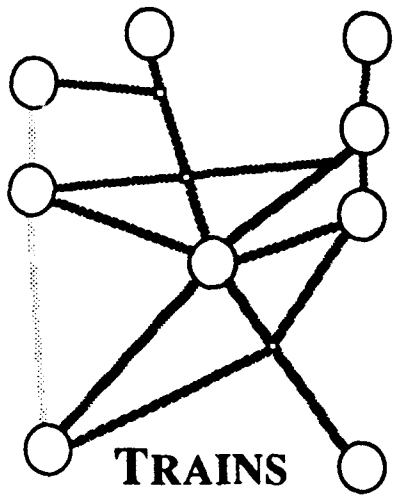
AD-A256 757



The TRAINS Project

DTIC
ELECTE
OCT 0 8 1992
S D

James F. Allen and Lenhart K. Schubert

TRAINS Technical Note 91-1
May 1991

92-26727

UNIVERSITY OF
ROCHESTER
COMPUTER SCIENCE

DISTRIBUTION STATEMENT A
Approved for public release;
Distribution Unlimited

The TRAINS Project[1]

James F. Allen and Lenhart K. Schubert
University of Rochester

Abstract

The TRAINS project is a long-term research effort on building an intelligent planning assistant that is conversationally proficient in natural language. The TRAINS project serves as an umbrella for research that involves pushing the state of the art in real-time planning, planning in uncertain worlds, plan monitoring and execution, natural language understanding techniques applicable to spoken language, and natural language dialog and discourse modelling. Significant emphasis is being put on the knowledge representation issues that arise in supporting the tasks in the domain. This report describes the general goals of the TRAINS project and the particular research directions that we are pursuing.

[1]This research was supported in part by Rome Lab contract F30602-91-C-0010, by ONR/DARPA grant N00014-82-K-0193, by ONR grant number N00014-90-J-1811, and by NSF grant IRI-9003841.

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: TR 382 and TN 91-1	2. GOVT ACCESSION NO.
3. RECIPIENT'S CATALOG NUMBER	
4. TITLE *(and Subtitle)*: The TRAINS Project	5. TYPE OF REPORT & PERIOD COVERED: technical report / note
	6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s): James F. Allen and Lenhart K. Schubert	8. CONTRACT OR GRANT NUMBER(s): N00014-82-K-0193, N00014-90-J-1811
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Computer Science Dept., University of Rochester, Rochester, NY, 14627, USA	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS: Defense Advanced Research Projects Agency, 1400 Wilson Blvd., Arlington, VA 22209	12. REPORT DATE: May 1991
	13. NUMBER OF PAGES: 39 pages
14. MONITORING AGENCY NAME & ADDRESS *(if different from Controlling Office)*: Office of Naval Research, Information Systems, Arlington, VA 22217	15. SECURITY CLASS. *(of this report)*: unclassified
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT *(of this Report)*

Distribution of this document is unlimited.

17. DISTRIBUTION STATEMENT *(of the abstract entered in Block 20, if different from Report)*

18. SUPPLEMENTARY NOTES

None.

19. KEY WORDS *(Continue on reverse side if necessary and identify by block number)*

planning; natural language understanding; dialog systems

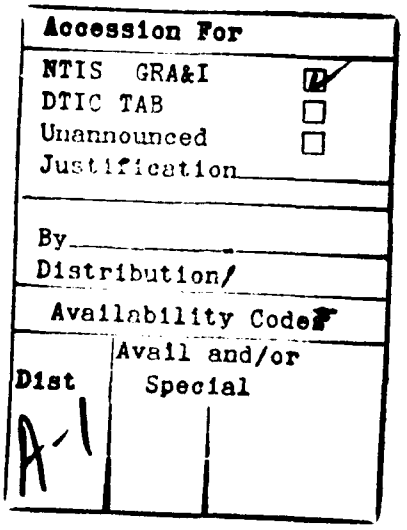
20. ABSTRACT *(Continue on reverse side if necessary and identify by block number)*

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

20. ABSTRACT

The TRAINS project is a long-term research effort on building an intelligent planning assistant that is conversationally proficient in natural language. The TRAINS project serves as an umbrella for research that involves pushing the state of the art in real-time planning, planning in uncertain worlds, plan monitoring and execution, natural language understanding techniques applicable to spoken language, and natural language dialog and discourse modelling. Significant emphasis is being put on the knowledge representation issues that arise in supporting the tasks in the domain. This report describes the general goals of the TRAINS project and the particular research directions that we are pursuing.



DTIC QUALITY INSPECTED 1

1. Introduction

The TRAINS project is a long-term research effort on building an intelligent planning assistant that is conversationally proficient in natural language. The name of the project comes from the domain used to test and demonstrate the ideas: the system acts as an intelligent assistant to a person (the **manager**) attempting to solve transportation problems involving freight trains and factories in a simulated world. The system assists in formulating plans, and monitors these plans as they are executed by the simulated agents in the TRAINS world, providing updates and support to the manager in replanning as necessary. The system does not have direct access to the (simulated) world, which is constantly changing as the (simulated) TRAINS-world agents, namely train engineers and factory managers, go about their tasks. Rather, the system must interact with these agents to find out information about the world, and synthesize this partial information into a coherent and reliable representation the the overall world.

The goal is to allow manager and system to communicate using unconstrained natural language and a graphics interface. Ultimately this will be spoken language, although at the start we are using keyboard input with no graphics support. The initial test input, however, is based on transcripts of spoken dialogs collected in a situation where a person played the role of the system. The input is thus full of false starts, ungrammatical sentences and other complexities not handled by systems that process written language. The dialogs themselves also exhibit complex behavior as both the manager and system take initiative at times in the dialog, and there are a large number of clarifications, corrections and acknowledgements in the dialog.

The TRAINS project serves as an umbrella for research that involves pushing the state of the art in real-time planning, planning in uncertain worlds, plan monitoring and execution, natural language understanding and natural dialog systems. Significant emphasis is being put on the knowledge representation issues that arise in supporting the tasks in the domain. The general research goals of this project fall into the following main areas:

- Parsing *real* utterances: Current parsers are not robust or tailored to the high level of "ungrammaticality" present in spoken language, which includes numerous sentence fragments, word errors, and false starts;
- *Interactive* man-machine dialog: which involves mixed initiative dialogs and a high incidence of discourse-level phenomena (e.g. clarification, acknowledgment, confirmation, correction, and so on);
- *Real-Time* plan reasoning: the simulated world is continually changing and the agents may have other goals besides what the system has sent them. If the planner waits too long before doing something, an opportunity to achieve a goal may pass; and
- *Plan execution and monitoring*: the use of causal and temporal reasoning, planning and plan recognition to effectively monitor a dynamic, uncertain world during a plan's execution, and replan as necessary to maximize the plan's chance of success.

The system can roughly be broken into three major clusters of subsystems. The language subsystem involves parsing, semantic interpretation, de-indexing (i.e. tense and reference processing), generation, and structurally-based discourse processing. The emphasis in the research is in dealing with real utterances and dialogs as they are spoken, full of restarts, sentence fragments

and a high level of discourse-level phenomena (e.g. clarification, acknowledgement, confirmation, correction, and so on).

The plan reasoning subsystem involves reasoning about both the dialog and the task domain. This includes plan construction, plan recognition, plan execution and plan monitoring at these levels. Remember that in TRAINS, execution is done by the TRAINS-world agents. As far as the system is concerned, executing an action involved dispatching the act to a TRAINS-world agent, and interpreting reports that come back from the agents to update the world model. When unexpected things happen in the world, the system must recognize whether it impacts the current plan or not, and must initiate re-planning with the manager if necessary.

The third subsystem is the simulated world. The TRAINS world is captured by a multi-level simulation. The physical properties of the world, including the actual actions that can be performed and external events that might occur, form the base for simulating the effects of actions on the world. Operating on this "world" are the TRAINS-world agents, the factory managers and train engineers, who actually "perform" actions in the world in accordance with the instructions that they receive from the planning system. These agents may have limited planning abilities themselves in order to create plans to solve typical low-level problems in the TRAINS world.

This technical report gives an overview of the project and serves as a general introduction to a series of technical notes on the various subparts of the TRAINS project. Updated lists of the available TRAINS reports will be published on a regular basis.

2. An Overview of the Initial Demonstration System

To start the project, we built an initial prototype system by adapting our previous research and applying it to the TRAINS world. While very limited compared to the eventual system, it does give a flavor for some of the complications that arise in the domain. In TRAINS-90, the dialog only concerned the specification of a simple plan, which was recognized and refined by the system and, when the dialog was completed, sent to the TRAINS-world agents and executed on the TRAINS simulator.

Consider two sample dialogs that currently run on the prototype system. These dialogs, while constructed, are based on transcripts of actual spoken human-human conversations in the TRAINS domain. As mentioned above, the current system only allows keyboard input. Figure 1 contains a fragment of the TRAINS world relevant to the dialogs. It contains three cities named I, B and G. There is an OJ factory at B, oranges at I and bananas at G. There is one train engine, E3, which is on route between cities G and I. There is an empty car at I, and E3 is pulling another empty car. The two dialogs are variants of each other depending solely on the manager's response to a clarification request. The input to the system is exactly as presented but the system responses are currently simulated by table look-up from the goals sent to a dummy "generation module". The discourse reasoning system, however, does generate conversational goals that could plausibly motivate the output.

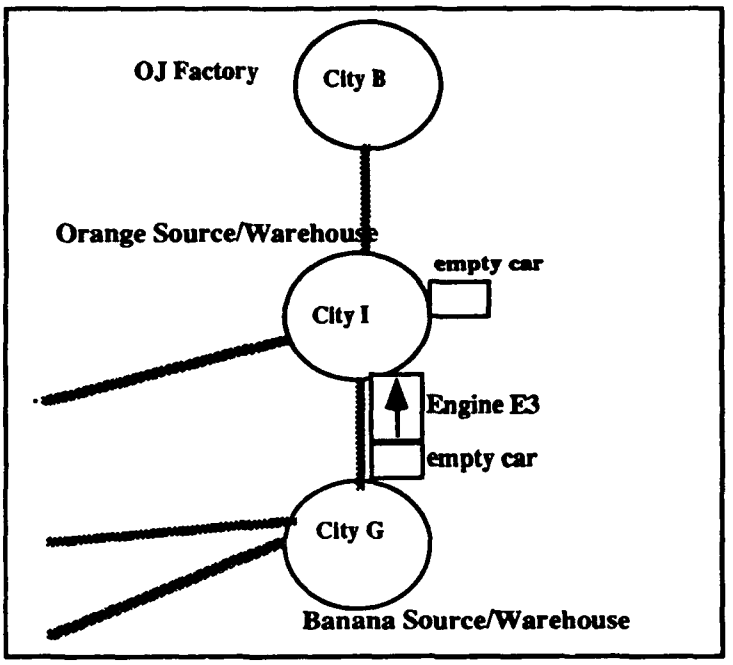


Figure 1: The domain fragment for the demo dialogs

The dialog follows. Actual input is in plain text and "canned" system output is in italics.

MANAGER: We have to make OJ.	(1)
There are oranges at I	(2)
and an OJ factory at B.	(3)
Engine E3 is scheduled to arrive at I at 3PM.	(4)
Shall we ship the oranges?	(5)
SYSTEM: *Yes,*	(6)
shall I start loading the oranges in the empty car at I?	(7)
MANAGER: Yes, and we'll have E3 pick it up.	(8)
OK?	(9)
SYSTEM: *OK*	(10)

In response to the first five utterances, the system is able to construct a plan to use engine E3 to ship the oranges at I to city B. But an ambiguity remains as to which car to use for the oranges. This problem generates a discourse goal to agree on a particular car, which is the motivation behind the systems response in (7). Since the manager accepts the system's proposal, the system can then fully specify an executable plan and transmit this to the TRAINS-world agents. In particular, the factory manager at city I is told to load the oranges in the empty car at city I before E3 arrives and to couple the loaded car to E3 when it does arrive. The engineer of E3 is told to go to city I, pick up the car and take it to city B. Finally, the factory manager at B is notified to unload the oranges and make OJ when E3 arrives there.

The alternate dialog is identical up to utterance (7). In this case, it continues as follows:

MANAGER: No, use the car attached to E3. (8)
SYSTEM: OK (9)

In this case the system's suggestion is denied, and an alternate plan is suggested by the manager in utterance (8). The system accepts this new suggestion and can then construct a similar plan to the one above except that the factory manager must wait until E3 arrives and then load the car attached to E3 with oranges. No coupling or decoupling is required in this second plan.

Figure 2 shows the overall architecture of the current system. The English input is first analyzed by a GPSG-style syntactic parser that produces a parse tree. The parse tree is then interpreted into an indexical logical form. This logical form strongly resembles the syntactic structure of the initial sentence and contains many forms of indexical terms that will need to be resolved in context. For instance, the indexical logical form does not indicate any scoping restrictions of quantifiers or certain operators, and does not resolve any referential expressions whose semantic interpretation requires a knowledge of context. For example, the sentence *We have to make OJ* would be parsed into the surface logical form

```
(PRES (SPEAKER TELL HEARER
      (THAT
            ((PRES MUST)
                  (WE (MAKE ( λx (x = (K OJ)))))))))
```

While this may look complex, it is simple to compute as it reflects the structure of the sentence and could be paraphrased (ignoring tense operators) as " The speaker tells the hearer that it is an obligation that they perform an action of type making something that is of kind OJ". There is clearly not the space here to justify all the details, but the interested reader should see Schubert and Pelletier [1982] and Schubert and Hwang [1990] for more details. This indexical logical form is then used as input to a module that scopes the quantifiers and operators, resolves the anaphoric referents based on the conversational context, and performs some automatic inference based on meaning postulates. The final output is a fully scoped representation in episodic logic [Schubert and Hwang 1989] as follows:

```
(∃e (e AT-ABOUT Now12)
      ((Hum TELL Sys
            (THAT (∃ e1 (e1 AT-ABOUT e)
                  (((K (λx ((SetOf Hum Sys) (MAKE OJ)) ** x))
                        MUST-OCCUR)
                  ** e1)))
      ** e)
```

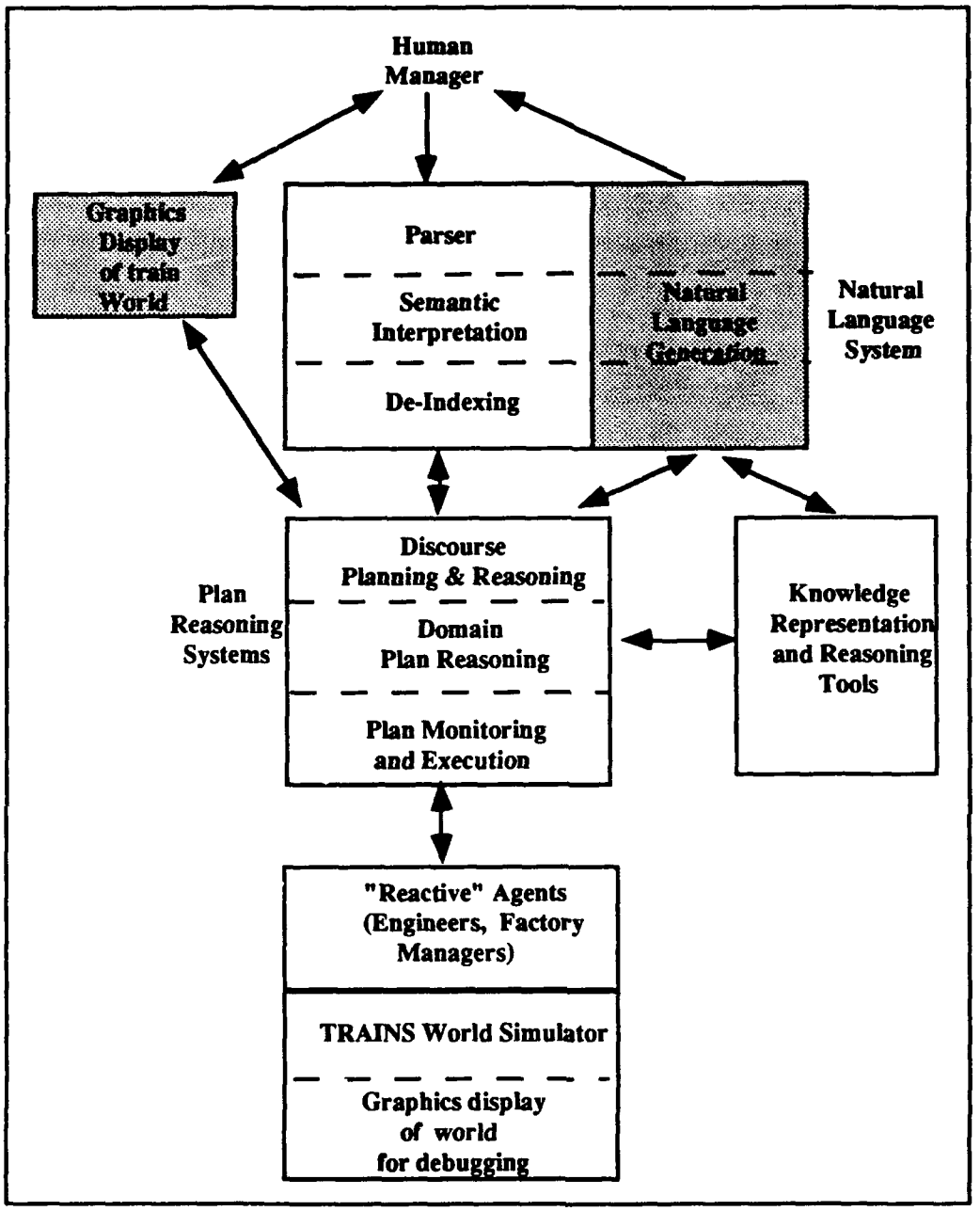


Figure 2: The TRAINS System Architecture

This form is then translated into a simplified representation used by the plan reasoning system. The input to the discourse reasoner is then the surface speech act

(TELL Hum Sys (Obligation SysHum (Make-OJ ?agent ?event)))

In the current system, all inform acts are taken to be suggestions about the plan that the two conversants are trying to construct. In particular, one may suggest goals to be pursued, or suggest steps in the plan, or suggest conditions relevant to the plan, or otherwise suggest general constraints on the plan. In this case, this sentence is clearly a suggestion of a joint goal (since it explicitly mentions a joint obligation). The discourse reasoner requests the domain plan reasoner to

incorporate this as a joint goal. Since no other goals are currently known, the domain plan reasoner finds the suggested goal reasonable and returns a success and updates the knowledge base that the manager has proposed the goal of making OJ. A person might acknowledge this goal at this time and make the goal a shared goal, or might wait for the next input. The TRAINS-90 system uses a very simple turn-taking strategy: the speaker retains the turn until a question or request is made, at which point the turn is given to the hearer. Since the first utterance is taken to be a suggestion, the system waits for the manager to continue speaking.

The next utterance, while appearing to be a simple inform (i.e., that there are oranges at I), must be interpreted by the system as a suggestion to use the oranges at I in order to make the OJ. Similarly, utterances (3) and (4) suggest which factory to use, and which engine to use to ship the oranges, respectively. Once the dialog is complete, the system has built an abstract plan that involves loading the oranges into the car at I, coupling this car to engine E3 when it arrives at I, moving E3 with the oranges to city B, unloading the oranges and starting production at the factory. This abstract plan is then decomposed into individual instructions for each of the TRAINS-world agents involved, namely, the engineer E3, and the warehouse manager at I and the factory manager at B. Instructions are in the form of condition-action pairs. For example, one of the instructions to the warehouse manager at I, W1, would be of the form

((AT C2 CITY-I)) -> (LOAD W1 C2 O1)

i.e. when car C2 is at city I, load it with the oranges O1.

This initial demo gives the general idea of the system but is currently limited in many important ways that form the focus of our current research. These can be divided into planning issues and language issues:

- **Planning Issues**
  - the present system constructs a complete plan before it sends off anything to the agents—the eventual system will need to be able to incrementally execute the plan while other parts are yet to be defined;
  - the present agents in the simulation do not have any other goals that would conflict with the system's plan—the eventual system must be able to handle conflicts and the system must decide whether to override the agents' other goals, or to renegotiate with the manager;
  - The present system starts with a complete representation of the world with regard to what trains and factories exist, where cars are, and so on—the eventual system will need to query the TRAINS-world agents in order to find out much of the information about the world.
- **Language Issues**
  - The present system uses standard parsing technology and deals only with "grammatical" sentences—the eventual system will deal with the fragmented utterances common in the collected dialogs;
  - The present system deals with typed input only—the eventual system would include both speech input and graphics-based interaction;
  - the present system uses a rigid turn-taking strategy—the eventual system will allow more natural turn-taking behavior and allow for interruptions;
  - The present system deals with simple forms of acknowledgement and clarification—the

eventual system must deal with corrections and modifications involving retracting information that has previously been mutually agreed upon, and correcting misconceptions that the manager may have.

This rest of this report describes the research issues that are of prime concern in the TRAINS project. These fall into three general areas: problems in plan reasoning and execution, problems in natural language understanding, and problems in discourse-level reasoning and dialog management.

3 Plan Reasoning, Plan Execution and World Monitoring

Most planning systems take a very narrow view of plan reasoning. In particular, they focus on plan construction - producing a sequence of actions that will achieve some specified goal. The representation of the world is very limited - typically a static world that changes only as the result of the planner's actions. In a more realistic scenario such as the TRAINS world, there are many different forms of reasoning related to plans. One is plan construction, but equally important is plan recognition, and using plans to predict future states of the world when the system's knowledge is incomplete. Furthermore, the plan-based reasoning is just one aspect of the more general representation problem of representing causal and temporal information in TRAINS. Thus we view planning as just one specialized form of reasoning within a more general system for representing and reasoning about the TRAINS world.

One of the crucial problems in managing plan-directed activity in realistic settings is knowing what state the world actually is in, and how that relates to the plan being executed. As important as this is, the techniques developed in the AI planning community have for the most part failed to address it. Traditional generative planning systems, for instance, can only operate with complete descriptions of the world. Typically, change occurs only as the result of the planning agent's actions, and the effects of that action can be computed precisely. In essence, planning in this traditional way is a reasoning exercise, and not related to acting in the world. These research projects have made progress in developing techniques for generating or adapting plans to situations, but have not addressed the knowledge representation issues concerned with executing plans with uncertain knowledge about a dynamic world.

In TRAINS, the world is constantly changing, so that the information the system receives tells it what the world was like in the past, not what is true now. The actions that the system plans may have unreliable effects, and the information the system receives may also be unreliable. How can such a system tell whether its plan is being executed appropriately? How can it tell where it is in its plan, given that most actions are being performed by other independent agents? Abstract reasoning about the scheduled time of events cannot help significantly; the execution of actions cannot be precisely controlled and unexpected situations may cause delays. Rather, the system must continually be matching the reports it receives about the world to the representation of what it expects in order to identify what part of the plan is being executed, and whether departures from expectations are serious (and require replanning) or are insignificant variations.

The TRAINS project will address a range of specific technical problems that involve plan reasoning in a variety of forms. In particular, we view the following as the key research problems:

- developing a representation of actions and plans that supports plan construction, plan

recognition, plan-based prediction, and other forms of reasoning using planning knowledge;

- developing a rich temporal-representation: we need to represent external events, continuous change, simultaneous actions, and the past, present and future in order to relate the plan-based expectations to the actual world;
- reasoning with highly partial, and possibly unreliable knowledge, both about the beliefs and intentions of the manager, and the state of the world;
- inferring what was true in the past, what is true in the present and what may be true in the future based solely on reports of the past;
- handling multiple levels of abstraction, both to allow for efficient plan reasoning about the external agents' (e.g. the train engineers) behavior, and to enable a reasonable man-machine interface.

There are many different parameters that affect the complexity of problems in the TRAINS domain. The ones that will be of most concern to us are the following: **Completeness**: do we have complete information about the world, or is it partial; **Reliability**: is the information we receive reliable; **Timeliness**: is the information we receive about the present, or a report about some situation in the past, or a prediction about the future; **Real-Time**: is the world changing as the system reasons, or does it remain static until the reasoning stops; **Complex goals**: do the plans involve specific goals that define success absolutely, or is there a set of utilities that capture tradeoffs between various goals; **Adaptability**: can the system respond to changing plans as the manager introduces new goals or revokes existing goals, and can the system diagnose and correct user errors and misconceptions; **Simulated-agent capabilities** - are the TRAINS-world agents performing the actions capable of planning independently, or do they simply "reactively" execute the actions they are told to perform.

The TRAINS simulator [Martin and Miller 1991] is designed so that we can vary the different sources of complexity mentioned above. In particular, we can start with a simulation where there is no cost to reasoning (i.e. time does not move forward while the system is planning), where world reports are accurate and complete, and where unexpected complications do not occur. We can then successively relax the restrictions - say by allowing partial knowledge about the world, then by introducing unreliability in the reports from the external agents, and then by introducing the real-time aspects - namely that the world continually changes while the system is reasoning.

Particular representation issues concerned with plan reasoning, plan execution and world monitoring are discussed in more detail in the rest of this section.

3.1 The representation of time and space

It is clear that the state-based approaches to representing the world (as used in STRIPS, NOAH, NONLIN, DEVISER, SIPE, etc) will not be adequate in domains such as TRAINS. In particular, these representations cannot handle extensive concurrency or uncertainty. Furthermore, the TRAINS domain requires an explicit model of time. There are two basic approaches to representing time, namely relative and metric time. Each has its advantages and scheduling domains will require both capabilities. In particular, there are specific deadlines by which time certain goals must be achieved, and these induce constraints on when planned actions can occur. A metric representation of time is ideal for this. On the other hand, there are other constraints that

cannot be captured by standard metric-based models. A train may need to be used for two different tasks, for instance, but it may be that it doesn't matter which task is done first. We would like to express that these two actions cannot overlap yet may be in either order. This cannot be expressed with current techniques using metric time for scheduling, except by enumerating the possible linear orderings. It can be expressed concisely, however, in a relative-time model such as developed by Allen [1983b].

One of the goals of this project is to develop a temporal representation that can accept and reason with both forms of knowledge. It is a simple matter to add metric time constraints to Allen's intervals and use them to initiate inference, but this is not a good long-term solution. In order to handle huge amounts of temporal information, the system needs to take advantage of metric information when it is available to represent the information more concisely than in Allen [1983b]. One interesting possibility is to partition the database into subparts, each one involving a linearizable order. Each partition can then be represented using metric methods. Miller and Schubert [1988] developed a system using this approach. The more general relative information would only be used when necessary.

The other crucial representation problem in these domains is that of space. As with time, most representations of space have been metric - co-ordinate based - treating the world as a grid. While this might be a possible representation for the scheduling domains, it is certainly not an efficient or convenient representation. In particular, there are specific constraints on how objects move through space. In our train-based world, trains can only move on tracks. Airplanes, likewise, fly fairly standard routes and are constrained to landing at airports. For most cases a graph representation of space, specifying the connectivity and distance information seems most appropriate. This representation uses an abstraction in the spatial domain - namely that of paths. Movement is restricted along certain paths, and paths may have certain properties, such as travel-distance (which may differ from absolute distance due to obstacles such as mountain ranges), or risk factors (say unreliability of the track, or hostile activities), and so on.

Reasoning about motion connects paths with time and there is a close correspondence between a path and the time it takes to traverse the path - temporal constraints can translate into travelling constraints and vice versa. We will be building a representation that allows us to reason about movement along paths through space and allow information about either movement constraints or temporal constraints to affect each other in the appropriate way.

As a parting word, it is important to realize that the system needs to be able to operate in situations where the complete state of the world is unknown. Any technique that depends on having complete information about every time and location will not be of use. The constraint-based representation of time, and the one we develop of space, is ideally suited for dealing with incomplete information and integrating new information as it is incrementally added.

3.2 Inference and causality

All planning systems represent causal knowledge about the planner's own actions - namely each action's prerequisites and effects. This information is used to compute the resulting state of the world when a given plan is performed. This information is generally specific to a particular reasoning task, usually plan generation, and is not directly usable for other tasks. For instance, the system might receive a report that a certain action (not necessarily part of its plan) just occurred.

What can it conclude about the world from this report? Standard planning models provide no answer. However, the temporal planner developed by Allen and Koomen [1983] and Allen [1991] represents all information about actions and events in a uniform temporal representation. In particular, knowledge about an action may include much more than simply the action's effects. For example, the fact that the action occurred may also introduce constraints on what the world was like just prior to it execution, as well as during the action's execution.

External events and actions by agents other than the planner are also not easily handled in traditional planning systems, yet are prevalent in the intended domain. An explicit temporal representation, however, allows a uniform representation across the planner's actions, other agent's actions and external events, and thus seems promising for integrating reports about these different activities.

To interpret a report fully, the system must perform more extended causal inference. For instance, consider a situation where train T1 needs to deliver half its cargo to destination A and the other half to destination B, and the plan does not specify in what order this should be done because either would be acceptable. When a report comes in that T1 has arrived at station C, which is on the path to station B, the system should infer that T1 will be going to station B first, followed by station A. Thus the system's predictions about the plan and the world state may change significantly based on a single simple observation. The techniques needed to do this are similar to those developed in work on plan recognition (cf. Kautz and Allen [1986]).

As a final point, it is important to realize that reports generally state what has already been done, or possibly what is expected in the near future, and rarely describe what is true in the immediate present. In order to predict the present state of the world, the system must be able to use causal knowledge to predict forward from reports about the past and backwards from reports about the future. Many of the models of temporal projection, such as those developed by Dean and McDermott [1988], and formal models such as Kautz [1986] and Shoham [1988] only allow prediction forward in time. Using the temporal logic directly, we can define a version of persistence that operates independently of direction (treating persistence as a form of non-monotonic equality between temporal intervals). An initial version of this technique has been developed in Allen et al (1991) and we expect to be able to extend it to the problems in the TRAINS domain.

3.3 Uncertainty

Uncertainty pervades all aspects of the kind of complex, ever-changing problem domain under consideration. The record of past events, the description of the current situation, anticipated future events, the external sources of information, and the internal general knowledge used for inference are all more or less unreliable. In such a setting it is crucial to work out the impact of uncertainty on (a) representation , (b) inference , and (c) control .

Representation

Consider a scenario in the TRAINS domain where there are many transportation tasks to be accomplished (with time and ordering constraints), and many different resources (engines, cars, supplies, other agents) potentially available to accomplish the tasks. A plan/schedule is assumed to be in effect already, so some of the tasks will be in a state of partial completion, and some of the resources will already be in use (or partially used up, in the case of depletable ones). In the later

stages of the project, new tasks and new resources may come on-line at any time.

In such a situation, allowing for uncertainty entails, for example, that the system can represent such probabilistically qualified facts as "Train#4 probably departed within 5 minutes of the scheduled time from station A"; "Train#4 is unlikely to encounter delays (such as loading delays at scheduled stops) along its planned route", and "Train#4 as likely as not will arrive at station B within 20 minutes of its scheduled arrival time, and almost certainly within an hour of its scheduled arrival time". Such statements attribute qualitative or quantitative probabilities to propositions .

However, the basis of propositional probabilities often lies in statistical probabilities (relative frequencies), such as that "Loading of tanker cars almost always finishes without a hitch", "The probability of a signal being overlooked or misread by the train engineer is higher near the end than near the beginning of the engineer's working day", or "Freshly harvested bananas survive a week's storage unspoiled in at least 90% of all cases." So these kinds of probabilities need to be representable as well, and their connection to propositional probabilities must be clearly understood. The work of Kyburg [1987] provides a good starting point for incorporating both types of probabilities into a general knowledge representation. In essence, Kyburg's statistical probabilities are of the form $P\{A|B\} \in [p,q]$, i.e., they provide bounds on the conditional probability of membership in set A, given membership in set B. The connection to propositional probabilities is as follows: the probability of "$b \in A$" is bounded by (p, q) if the conditional probability $P\{A|B\}$ is bounded by (p, q) and there is no subset B' of B such that bounds on $P\{A|B'\}$ conflicting with p, q are known. Bacchus [1989, 1990] generalized the representation of statistical probabilities and their connection to propositional probabilities, with first-order formulas replacing Kyburg's sets. Loui [1987] and Halpern [1989] provide some closely related studies.

It seems clear that simple numeric probability bounds will ultimately be inadequate, since the probabilities of certain outcomes changes systematically with the values of certain parameters. For example, the earlier rule about spoilage of bananas was quite crude, and what one would really like to say is that the probability of remaining unspoiled declines monotonically with storage or transit time, according to a formula such as $e^{-\lambda t}$; (cf. Dean and Kanazawa [1988]) Another simple example of this type is that the likelihood of a train derailing in a turn may well depend in a quantifiable way on the speed of the train and the curvature of the track, and perhaps other factors; at the very least we want be able to say that the faster the train and the tighter the turn, the greater the risk.

Inference

Uncertain knowledge leads to a style of reasoning often called "evidential reasoning". The work of Kyburg and the other authors cited above is very much concerned with evidential reasoning in the sense of making justifiable (or at least reasonable) probabilistic inferences about particular circumstances based on those circumstances and on statistical facts and other general knowledge. However, the work does not yet provide a general foundation for "combinatory" probabilistic inference, i.e., probabilistic inference based on any number of uncertain facts and unreliable rules. Yet such combinatory inference is extremely important in the kind of plan-based reasoning under consideration. For instance, the above prediction that "Train#4 will almost certainly arrive at station B within an hour of its scheduled arrival time" could depend on many more or less certain facts and rules: that according to the current plan, the tracks to be used by the train are unlikely to be

preempted by other trains, requiring sidetracking; that accidental track obstruction, derailment, or engine trouble are extremely unlikely under the circumstances expected; that loading/unloading delays are unlikely for the scheduled loading/unloading stops; etc.

Some aspects of this general problem which need to be investigated are "parallel" and "serial" combination of information, and the avoidance of "zig-zag" or "circular" reasoning. Parallel information combination involves the use of different lines of inference bearing on the same conclusion. For instance, suppose we are interested whether a particular truck-delivered shipment to be loaded onto a train will be available as scheduled. There may be an argument based on the "track record" of the shipper that the delivery will probably be on time, and another argument based on weather conditions that it will not. Somehow, a "compromise probability" (or probability interval) needs to be inferred. In other cases the parallel arguments may reinforce each other, and should raise the overall probability. A key issue here is that of independence : clearly, parallel arguments should reinforce each other only if they are (in some appropriate sense) based on independent evidence.

Serial information combination involves "chaining" forward from uncertain premises using unreliable general (e.g., statistical) and particular facts. Intuitively, certainty should attenuate with inferential distance. But here, too, it is unclear how to implement this in a defensible way, or how to take account of dependencies among uncertain facts used in a chain. The circular reasoning and "zig-zag" fallacies arise from neglecting such dependencies. Suppose, for instance that a train is headed from A to B, with a loading stop along the way. It is likely that the loading stop will proceed without a hitch, but there is some possibility that it will involve a delay, and also some possibility that it will not occur at all (if the load is unavailable). We might reason that since the loading stop is quite likely to proceed without a hitch, the train is quite likely to arrive at B on time. But if a train arrives on time, it is in general quite likely that any loading along the way either occurred as scheduled or not at all; and (proceeding circularly) this increases the original probability of the loading stop having occurred as scheduled; or (proceeding in zig-zag fashion) it increases the probability of the loading stop having been cancelled! Pearl [1988] offers partial answers, but only (in essence) in a propositional logic setting, which is expressively inadequate for present purposes. It is also worth noting that a large amount of work in "expert" (or "knowledge-based") systems has been nominally concerned with the combinatory inference problem. However, most of this work has been in the nature of providing efficient tools and environments for rule-based programming, with little concern either for the meaning of rules or the formal foundations of the combinatory methods provided. Nevertheless, we expect to derive some insights and techniques from that literature.

A crucial consideration in combinatory evidential reasoning is the need for belief revision . The system's knowledge about the world does not accumulate in monotonically increasing fashion, but rather may involve retraction as well as addition of information.

For instance, a previous report by one of the agents (or an inference from such a report) may be contradicted by another, later report; e.g., a shipment guaranteed to arrive according to an earlier report may fail to show up, according to a later one. The question is not only how to adjust the probability of the now dubious conclusion, but how to "distribute blame" among the (more or less certain) beliefs that led to it. For example, the non-arrival of an expected shipment might cast doubt on the assumption that it was ever loaded onto the designated train, or on the assumption that the

shipment was not unloaded at an intermediate stop, or on the assumption that the designated train has arrived. Which assumption is weakened the most depends on their certainty and how they, in turn, were derived.

We see at least two promising directions for further research here. One concerns the non-occurrence of expected events, the other the occurrence of unexpected events. For the former, we propose to view probabilistic belief revision as a generalization of non-probabilistic "truth maintenance systems" (e.g., Doyle [1979]). This involves keeping track of where conclusions came from, via dependency records. These dependency records provide candidate "culprits" when a proposition is contradicted. Even when the rules used for inference are statistical truths and the facts fed into them uncertain, it appears possible to provide plausible methods of propagating belief updates backward based on probability theory. Moreover, the dependency information also allows avoidance of circular and "zig-zag" reasoning.

The second promising direction involves the notion of "explanation closure" proposed in Schubert [1990] as a technique for solving the frame problem monotonically. Explanation closure axioms provide complete sets of alternative explanations for some type of event, and thus allow inference of the possible explanations from the occurrence of the event. A probabilistic version would give "virtually all" plausible explanations. For instance, a set of alternative explanations for a train derailment, at least one of which is almost certain to hold, is that the train went at excessive speed around a bend, that it went over a mis-set track switch, or that it encountered track damage or an obstruction. Even though one can conceive of other explanations, they can be discounted for practical, probabilistic inference. In a particular instance, the system may have strong reasons to believe that the derailed train was on an unobstructed, undamaged, uncurved section of track; it may also know that the train went over a switch just as it derailed, and not be very sure, a priori , whether the switch was properly set (e.g., probability >= .75). Then the probabilistic closure axiom would allow the system to conclude with considerable certainty that the switch was not properly set, contrary to the previous belief. So closure reasoning can provide a type of belief revision that somewhat resembles truth maintenance reasoning, but does not require explicit dependency information.

Control

In the early years of the project, the time taken for inference will be a relatively unimportant concern (since we will assume that reasoning cost is not traded off against other costs). There is an interesting analogy between the behavior of the system in the first years of the project and the behavior of a game-playing program without time constraints (or any other type of program that tries to make an optimal decision based on a tree search and node evaluation.) The system can be said to rely on "deep search" of the consequences of its actions (commands and requests to the TRAINS-world agents) and of events in the simulated world, with little, if any, guidance from "static evaluation" or "shallow search" for deciding where the most promising directions for deeper search might lie.

Eventually, however, the amount of reasoning that can be done will compete with other actions that the system can perform. In real-time reasoning, inferences cannot be pursued "at leisure", but only in a competitive, time-constrained fashion. There must be criteria for judging when an inference task is sufficiently important and urgent to be pursued, and when the conclusions reached

are firm enough and specific enough for the purposes at hand. Inevitably, such criteria will involve use of cost/benefit knowledge; clearly an inference on which human lives may hinge deserves more time and care than one with only some small economic consequences.

A conceptual difficulty that needs to be confronted here is that one seemingly cannot in general know how important an inference will be until one has made it. Individually innocuous facts may have startling consequences when appropriately combined. Imagine two tracks running in parallel and connected by a spur. On each track there is a train approaching the spur, from opposite directions. The switches are set so that the first train will move via the spur onto the second track; however, by the time it does, the second train should have passed the spur, so there is no problem. But now suppose that the second train makes an unexpected stop, and so the first train will reach the spur first. What is likely to happen next? Note that quite a few facts have to be brought into the reasoning process to foresee the imminent collision; and of course, one can easily think of situations where the significant future event implied by current circumstances is still further removed from those circumstances than in this example, in terms of the requisite inferencing.

Here again, the game-tree analogy seems helpful. What one wants is "rough and ready" methods of anticipating important consequences quickly, albeit unreliably, i.e., the analog of static evaluation or shallow search in game playing. This translates into having methods of inference and action at various levels of abstraction, both "coarse-grained" and "fine-grained". For instance, coarse- grained methods might include a rule that predicts a possible collision just on the basis of two trains being somewhat near each other, and in motion. It would not make the prediction with high probability, but given the high cost of collision, this would be enough to justify a more "fine-grained" look at the situation the two trains are in, so as to confirm or disconfirm with greater reliability that they are on a collision course. (Naturally, if the coarse-grained method also anticipates that there may be almost no time left before the collision, the wise choice may be emergency action rather than further thought.)

3.4 Abstraction[2]

The system must not only be able to represent what the world is like now, and what the plans are, but must also be able to determine how much of the plan has been successfully executed with regards to the given set of goals. In order to answer these questions, the system will be required to represent and reason about how each of its tasks and subgoals are related to the achievement of its larger goals. In addition, the system will often be required to tradeoff the increased accuracy associated with reasoning about plan details against the computational cost of doing so, and hence will specify plans for the autonomous agents (train engineers and warehouse managers) at different levels of description. Further, because of the inherent uncertainty of the system's information, reports of current world state may diverge from the system's predictions, and hence, previously pertinent subtasks may need to be abandoned or replaced without disrupting the entire matrix of other unaffected subtasks currently or soon to execute. The use of abstraction, in particular, encoding the partonomic (the way in which tasks subdivide into smaller tasks) and taxonomic (the alternative ways in which a task can be specialized) structure of a plan will figure prominently in our overall system design. Our previous research in plan recognition [Kautz and Allen 1986];

[2]Josh Tenenberg contributed much of this section on abstraction.

Kautz [1987] and abstraction [Tenenberg 1988] provides a hierarchical plan representation which explicitly represents the dependencies of the different actions in a plan. Plans are sets of actions subject to given temporal constraints, (A before B, C during E, A after C, etc.) where some of the actions are decompositions (or substeps) and some actions are specializations of others.

This structured approach is crucial in order to

1. encode the justification for each task in the plan with regards to the overall goals, allowing the system to reason locally about plan revocation, failure and repair.
2. pursue a least commitment approach to resource management, allowing for the deferral of specific resource choices until execution time,
3. provide information at an appropriate level of representation for interacting with the manager.

Further, the appropriate level of abstraction that is used by the system will often depend upon the level at which statistical information is available. Each of these aspects will be discussed in turn.

Hierarchical Goal Structures

The goals of the system will arise at different levels of abstraction. For instance, if the system has a goal of getting 10K of bananas from city G to city A, then it might have subservient goals of getting car 7 from city I to city G in order to affect the transfer of materials, which in turn requires getting engine 3 from city B to city I, setting track switches, etc. In problems of this nature, it will be crucial that the system have explicit encodings of the goal/subgoal dependencies of the different steps of the plan, so that strategies of least commitment can be employed, and so that plans can easily be repaired following either plan revocation or failure. Without such an encoding, it is conceivable that a large plan containing thousands of individual train movements would fail due to the revocation or failure of a small subpart of the plan, such as engine 3 being currently used for another, non-preemptable task. One of the hallmarks of intelligent behavior is the understanding of the relationship between each subtask and the larger tasks and goals to which it is subservient.

Figure 3 provides a sample fragment of a plan hierarchy for moving cargo from one location to another. The single line edges represent substeps, and the broad grey edges represent specializations. For example, moving cargo is achieved by moving a train car to the cargo, loading the cargo, moving the car to its destination, and unloading the cargo. Moving a car can be specialized to MoveBoxCar, MoveFridgeCar, or MoveTankerCar, depending upon the type of car that is available and/or required.

This plan structuring permits the adoption of a simple strategy for certain kinds of plan augmentation by the manager. In particular, if the plan addition it selects is a specialization of an already existing action, this can be handled by descending the plan hierarchy. For example, given the initial goal of getting 10K of bananas to city G, the manager might specialize this by adding a requirement that a box car must be used, resulting in a MoveBoxCar being used instead of the more general MoveCar action.

Deleting part of a plan can additionally be facilitated by the hierarchy. If the manager requests a deletion of a subpart of the plan, the system can ascend the hierarchy to either infer (or query the manager) whether superior goals will continue to be pursued or will be abandoned. If pursued, then sibling specializations or decompositions can be employed. For example, given the above plan

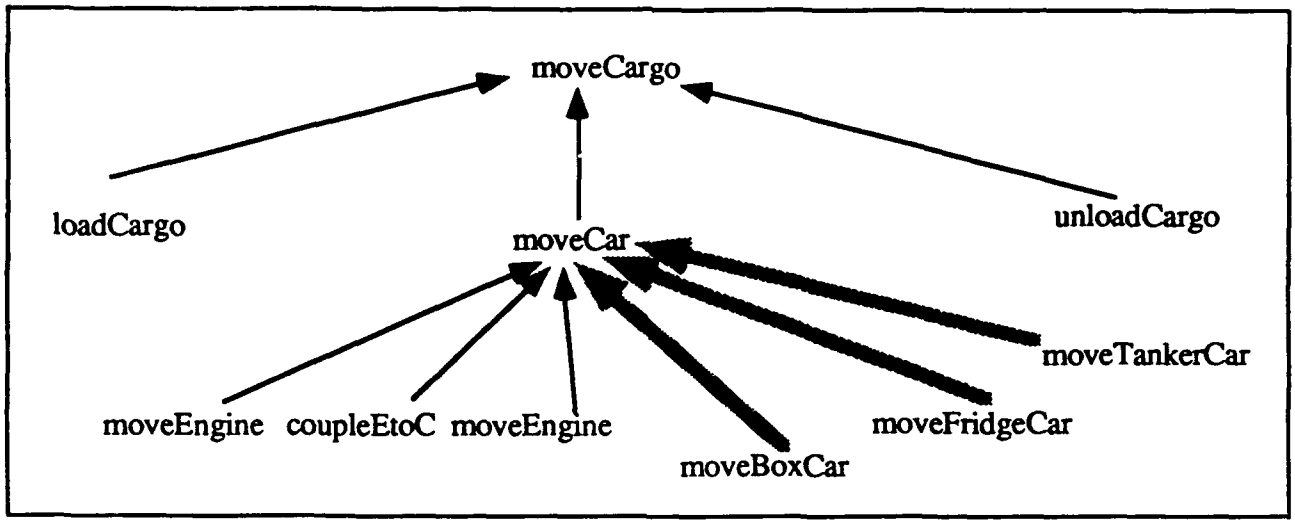


Figure 3: A Simple Plan Hierarchy fragment

addition of using a MoveBoxCar, the manager might decide that this comes at too high a cost, since the only boxcars are far away. By removing this constraint, the system is free to pursue the use of any available train car.

Simple plan repair will likewise be handled in a similar manner. For instance, if the system determines that a box car is unavailable when it was required, the system is able to infer, by ascending the hierarchy, that this car is being used to transport bananas, and that another train car can equally satisfy this goal.

Least Commitment

Because recovering from unexpected failures such as the one above exacts a computational cost, it would be best to develop strategies to avoid them. One such strategy is to defer the choice of resources as late as possible. For instance, given the same goal as above (getting 10K of bananas to city G), if the system were to defer the choice of type of train car until execution time, then the above problem might be avoidable since the availability of many resources changes rapidly over time. Using abstraction, one can construct plans using a generic train car without specifying the particular car type. Not only can one generalize by replacing specific object constants by variables, but, using our encoded object taxonomy, one can specify the most general class of object for the given task.

The intuition behind why least commitment is beneficial in the above example, is that the larger a set of cars that one considers for carrying the cargo, the greater the likelihood that the plan will eventually succeed, since it only requires that some car within the candidate set be available, rather than a particular car. The statistical representation we will be using as described earlier allows this intuition to be explicitly encoded. For instance, suppose the likelihood that any particular car is available at a given time is relatively small, since train cars are highly utilized, but that the likelihood that some car is available is quite high. Without the use of abstraction, one might abandon easily satisfiable plans, believing falsely that they are unachievable. Without the use of statistics, however, the gains obtained by planning at the abstract level are more difficult to quantify. This will ultimately enable the system to choose the appropriate level of abstraction with

which to plan a task in advance of its execution.

4 Natural Language Processing

To set the stage for our discussion of natural language processing (NLP), we begin with a brief characterization of the state of the art, and our approach to overcoming current limitations. We will then elaborate a little on the three input processing stages shown in Fig. 2 (syntactic parsing, semantic interpretation, reference and tense interpretation, plus a input-driven inference stage not implemented in the current system), and then describe the research issues associated with each of these stages, in the context of the TRAINS project. Many further details of our methodology and goals can be found in the technical reports on the TRAINS project ([Schubert 1991]; [Light 1991]).

4.1 State of the Art in Natural Language Processing

It is currently possible to analyze quite large subsets of English syntactically (e.g., Briscoe et al [1987]; Harrison and Maxwell [1986]; Sager [1983]; Jensen and Heidorn [1983]). These systems have quite substantial coverage of syntactic structure and large lexicons. However, current parsers

- are generally unforgiving of errors and thus inapplicable to real discourse, and
- are apt to give numerous alternative analyses of complex sentences (most of which are intuitively irrelevant), thus placing an unreasonable burden on subsequent interpretation and disambiguation processes.

In semantic interpretation, the situation is much worse. Only *very* small subsets of English can be analyzed semantically, for the following reasons:

- The meaning representations employed are expressively inadequate to handle such pervasive phenomena as adjectival and adverbial modification, generalized quantifiers, tense, aspect, temporal adjuncts, causal relations, propositional attitudes, and various types of nominalization. This makes it impossible to express many intuitively simple ideas, or leads to cumbersome circumlocutions that are hard to compute, understand, and use.
- The mapping from syntax to semantics is often very complex and "hidden" in procedures specific to words, word categories and types of syntactic fragments. This makes extension increasingly difficult - and brittle - as the size of the rule set grows.
- The role of *context* is poorly understood, so that it is extremely hard to get from a context-dependent ("indexical"), ambiguous logical form to a context-independent one, usable in concert with other knowledge in the system's (context-independent) knowledge base.
- The role of *inference and plan reasoning* is poorly understood, although they are known to be crucial to arriving at the intended interpretation of an utterance, and at a coherent, appropriate response. Part of the problem is again that most meaning or knowledge representations are too weak and rigid even to express the knowledge needed for inference and plan reasoning (e.g., meaning postulates, unreliable world knowledge and knowledge about individuals' intentional states).

We believe that a great deal of headway can be made on these problems using

- error-tolerant, preference-seeking parsing;
- close coupling of syntactic and semantic theory development;

- a "natural" meaning representation well-attuned to the expressive resources of natural language, and easily computed from surface syntax;
- formally well-founded, modularized, compositional rules of interpretation, and formal semantics for the meaning representation;
- a "divide and conquer" approach to the very complex transduction from surface form to genuine understanding of an utterance, breaking the process into relatively simple, conceptually independent, formally analyzable stages;
- experience with real discourse in a realistic domain; this will help us avoid the temptation to focus on the easy problems, rather than the important ones.

4.2 The NLP Architecture

We now give a slight elaboration of the stages of understanding as we conceptualize them in keeping with our divide-and-conquer strategy (cf., the first three boxes in Fig. 2). Figure 4 illustrates the conceptual subdivision of the understanding process with the example input, *A train pulled into the station.* This is presumed to be part of some larger discourse; for instance, the preceding sentence might have been *Passengers crowded onto the platform.* The first three compartments in the figure correspond to the first three boxes in Figure 2. The first stage, syntactic parsing, produces phrase structure trees like the one shown (except that syntactic features have been suppressed).

The next stage, logical form (LF) computation involves two phases, namely (a) production of a (very English-like) *unscoped* LF, one in which the exact scope of operators like $\exists$ (there exists), "past", and "The" is still indeterminate; this indeterminacy is signalled by the angle brackets; and (b) use of a heuristic algorithm to "raise" these operators to specific pre-sentential positions, with variables taking the place of the original angle-bracketed quantified expressions.

Even after scoping of operators, the resultant LF is logically indeterminate, since for instance "the station" has no specific meaning except in relation to a context, and similarly "past" has no specific meaning except in relation to a particular context of utterance (specifically, the time of utterance). So in the third stage a specific context structure called a *tense tree* is brought into the picture, to help eliminate the remaining indeterminacy in meaning. The notion of a tense tree [Schubert and Hwang 1990a,b] was originally developed to deal with tense, aspect, and time adverbial interpretation, and is being modified for the TRAINS project to support reference interpretation as well.

Finally, Fig. 4 shows a stage of input-driven inference, not currently implemented in TRAINS (and omitted in Fig. 2), which could derive immediate consequences entailed or suggested by the input. For instance, consider the following variant of the sentence under consideration: *A modern kind of high-speed train pulled into the station.* This sentence is literally about a *kind* of train, not about a particular one, but the existence of such a particular train is an obvious entailment. This would be obtained through a meaning postulate (MP) which says that if an object-level non-stative predicate (like *pulled into the station*) is applied to a kind of thing, then it applies to some *instance* of that kind. Inferences based on MPs often convert the implicit content of a sentence into a more explicit, useful form. Another type of example is the inference that the episode of passengers crowding onto the platform probably *immediately precedes* or *overlaps* the episode of the train arriving, and various other probable conclusions like the ones indicated. These sorts of inferences

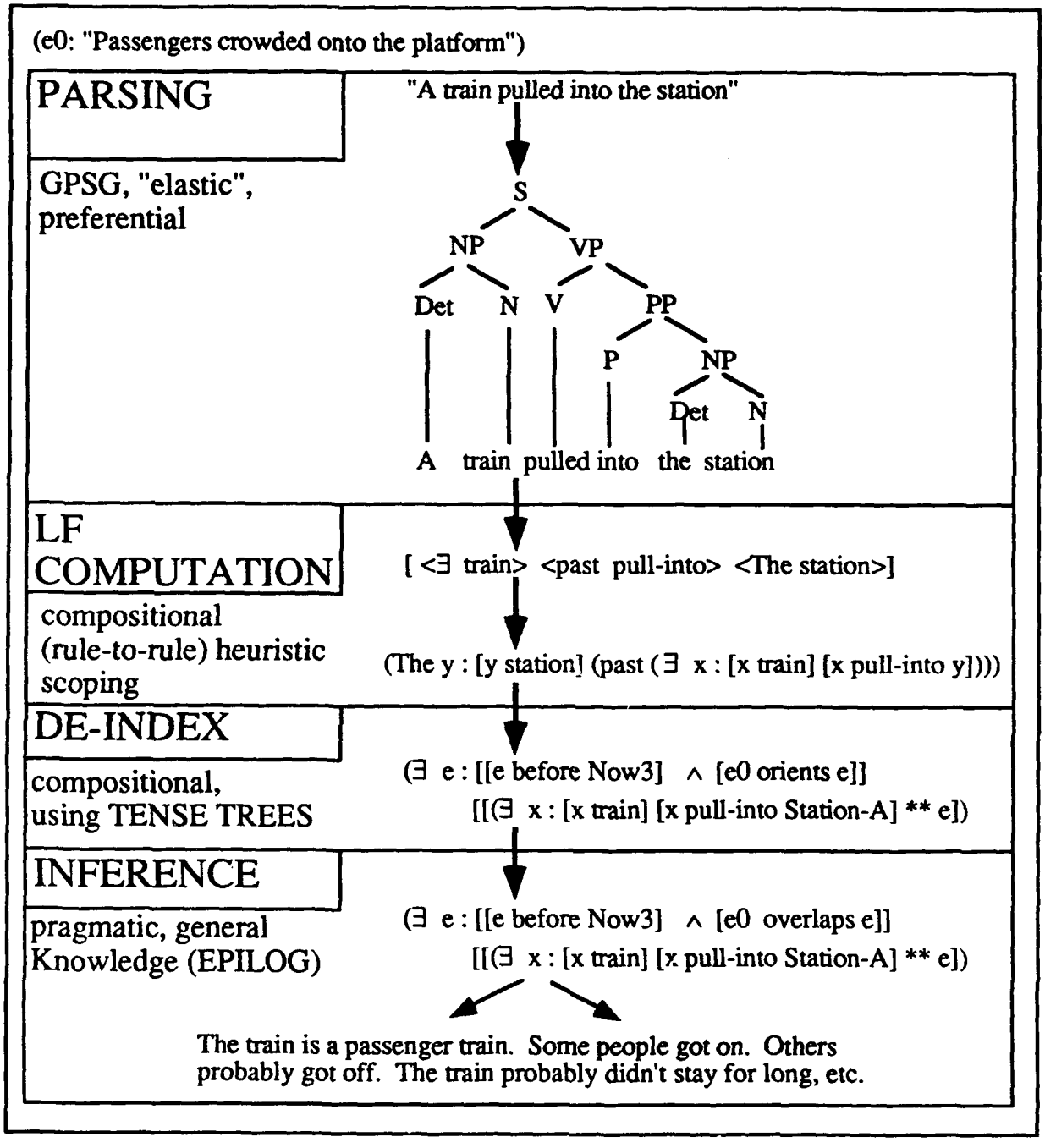


Figure 4: The (conceptual) stages of understanding

are currently handled in the EPILOG system [Miller, Schubert and Hwang 1990] and are expected eventually to play an important role in the TRAINS system, serving to bring to light entailments, to link new utterances temporally and causally to previous ones, and contributing to reference interpretation and more generally to the sense disambiguation process.

4.3 Parsing and Error-Tolerance

The parser we are developing is designed to be error-tolerant and preference-seeking. One key to

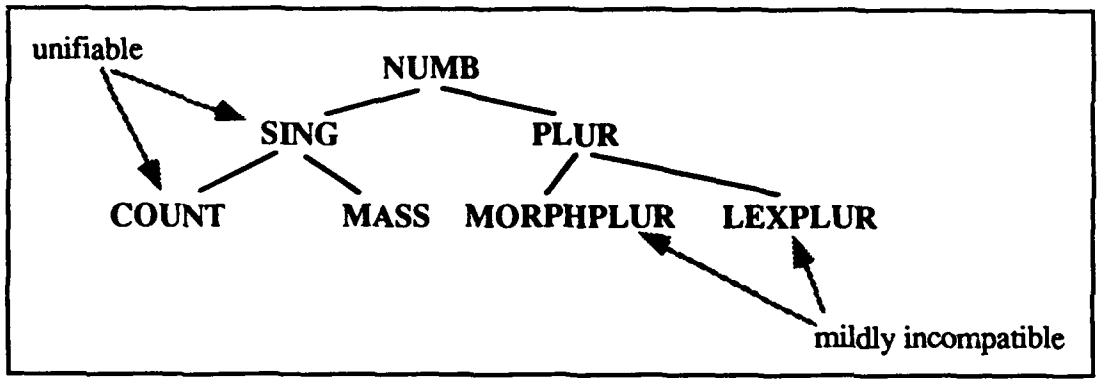


Figure 5: Elastic unification in type hierarchies of features

error tolerance will be a kind of "elastic unification" based on feature hierarchies such as the NUMB (i.e., number) hierarchy shown in Fig. 5. When the parser unifies two features on the *same* path, such as COUNT and SING in the figure, the result is an error-free unifier, namely the more specific of the two features -- in this example, COUNT. When it unifies features on *divergent* paths, such as MORPHPLUR (morphological plural) and LEXPLUR (lexicalized plural), the result is a (more or less severe) feature clash, but this is nevertheless accepted by the parser. As a result, the parser would tolerate a phrase like *the oranges shipment*, even though morphological plurals (like *oranges*) are generally discordant in prenominal position. Note that *cattle shipment* is impeccable, as *cattle* is a lexical plural. The parser would also tolerate more severe clashes, as in *The train arrive at the station*. On our account, this is judged as more severe because it involves diverging paths higher in the feature hierarchy, namely divergence at the root of the paths to SING and PLUR. Besides elastic unification, we plan to use a variety of additional techniques to attain the kind of comprehensive error-tolerance needed in the TRAINS domain. We will enumerate these after completing the description of the basic parser operation.

The "preference-seeking" character of the proposed parser entails that it should prefer certain syntactic and semantic choices and combinations to others, and discard inferior choices and combinations early on, so as not to be carrying along numerous alternatives at any time. Briefly, this preference-seeking behavior will be achieved by (a) building syntactic sentence structures that are as nearly complete as possible, given the words seen so far; (b) in these structures, propagating inhibitory and excitatory *potentials* corresponding to semantic and syntactic preferences; and (c) discarding all but the most highly activated global analyses "on the fly", retaining only 1-3 analyses at any point and typically ending up with only one complete sentence analysis. Also, as a method of preventing over-prediction of highly productive constructions such as coordination, we will allow for *delayed triggering* of phrase structure rules. E.g., coordination rules will be triggered by the coordinator or comma following the first coordinated constituent, rather than by that first constituent.

The preference-seeking strategy was described in Schubert [1986], and major parts of the parser have been implemented. However, it remains for future work to explore certain important details of the parser design, such as parts of the feature propagation mechanism, the morphological analyzer, the generation of semantic potentials, and how magnitudes of syntactic and semantic potentials can be estimated in a systematic fashion.

The current TRAINS parser does not use elastic unification or propagation of potentials. Crucially, however, it does employ feature hierarchies like those described above, and the grammar for the sample discourse is formulated in terms of this feature system. Also the parser and grammar allow optional and iterated (Kleene-star) constituents. Thus the grammar and lexicon will only require minor modifications in the change-over to the eventual error-tolerant, preference-seeking parser. This is important since we have found grammar and lexicon development to be even more time-consuming than parser development, even for a small lexicon. Also, the current parser does allow for variable syntactic rule potentials, and these are used to prioritize the order in which rules are tried. As well, it does incorporate feature agreement and propagation principles, namely versions of the GPSG Head Feature Principle and Foot Feature Principle; this obviates the need for explicit rules equating features of mother nodes with those of daughter nodes, and for separate mechanisms for unbounded dependencies. Also realization of agreement and subcategorization principles is facilitated by the grammatical representation, though not fully automatic (i.e., phrase structure rules must be explicitly annotated with agreement equations for sets of agreement features, and rules which combine verbs (etc.) with subcategorized constituents must be explicitly supplied). It is unclear as yet whether agreement and subcategorization principles in the final parser design will be built-in or specified rule-by-rule. This depends very much on the direction our grammar design takes as we try to accommodate more of the TRAINS dialogs and gain better empirical insight into the grammar design/parser design trade-offs.

In the rest of this subsection, we enumerate additional kinds of errors (besides concord errors) we plan to handle eventually. We expect to handle most of the phenomena by introducing new phrase structure rules and lexical rules (often with inhibitory potentials, i.e., costs), with minimal change to the parser itself. The reason for this expectation lies in two of the assumed parser features: delayed triggering of certain rules and on-line pruning of "non-preferred" alternatives. Delayed triggering can be used to curtail over-prediction of phenomena like restarted phrases, and pruning of non-preferred alternatives ensures that the parser will quickly give up predictions of ill-formed input when (syntactically and semantically) well-formed alternative analyses exist. Thus the parser will not entertain a large number of ill-formed analyses, as an all-paths parser would (given the same phrase structure rules for ill-formed input). An approach that allows for errors in the normal course of parsing is preferable to standard methods of "post mortem" error correction (e.g., see Allen [1983a] and Mellish [1989]). For instance, post mortem error correction would fail to detect an error in "I guess its time to quit" (whose most natural reading substitutes *it's* for *its*), since the uncorrected sentence is perfectly grammatical on a reading of *its time to quit* as a noun phrase. See Schubert [1991] for further discussion.

Semantic "errors" such as selection restriction violations (*My car drinks gasoline, Time flies*) are not errors for us at all. Rather, when a particular syntactic analysis corresponds semantically to a familiar type of predicate-argument combination, a positive (excitatory) potential is generated for that syntactic analysis. Consequently, it is more likely to "win out" over competing alternatives. When a selection restriction is violated, the result is typically an *unfamiliar* type of predicate-argument combination, which will not generate a potential and may therefore "lose" to alternative analyses. But if there are no alternatives, the unfamiliar combination is readily accepted. Moreover, even violated restrictions may generate excitatory potentials, if the violation is of a familiar, "habituated" type. For instance, the application of a verb of motion to a temporal argument is quite

familiar (near-idiomatic), as illustrated by *Time flies, The hours creep by, The days rush past*, etc. This, for us, is part of the reason why the imperative reading of *Time flies* is ordinarily not even noticed.

There are a number of phenomena on the borders of grammaticality that we will allow for. The most important of these is ellipsis, or rather a variety of types of ellipsis. These do call for modification of the parser, not just the addition of new phrase structure rules, namely the incorporation of a conservative strategy for postulating null constituents, when the parse state suggests that ellipsis may be present. (This is not a radical modification: the current backtrack parser already contains simple mechanisms for postulating null constituents and traces.) The major types of ellipsis are (i) null-headed NPs (e.g., *those five, the first, whose, few, the wealthy*); (ii) phrasal answers to questions, and other phrasal utterances (e.g., *engine E3*, in answer to *What engine is available?*); (iii) VP and predicate ellipsis (e.g., *Shall I notify the factory or will you?; I will, Is it or isn't it?*); (iv) conjunction reduction/gapping/stripping (e.g., *There are oranges at I and an OJ factory at B, Tanker t3 is ready to be filled and t4 emptied*); Sag et al. [1984] suggest an analysis which allows arbitrary phrase sequences in the second conjoin, but this is over-productive, and we instead have in mind a rule for recursive head deletion to account for this type of ellipsis; (v) pronoun, auxiliary, or determiner ellipsis (e.g., *Told you so, You hungry?, Trouble is, ..., Pity he's sick, Hi, you there?, How much OJ currently at C?*).

An important and challenging phenomenon in some of the dialogs we have collected are complex sentences uttered interactively, in "instalments". For instance, in one dialog (see the first excerpt in section 5.3), the manager utters *What if we would stop it uhh*, and follows up with three further clauses. These additional clauses still belong to the same sentence (outlining a plan), but each is followed by an acknowledgement by the system. The significance of this phenomenon for discourse modelling is discussed in section 5.3, but it certainly raises problems for parsing and interpretation as well. One might be tempted to regard the individual instalments as elliptic since they are sentence fragments, but the difference is that they are not to be "completed" by positing null heads or making structural analogies with preceding utterances, but rather by stringing them together. Thus the parser and semantic interpreter will also have to proceed in instalments, gradually building up a "complete" meaning.

Five other borderline phenomena are lists, parentheticals, telegraphic language, proscribed colloquialisms, and deviant vocabulary. Again, we expect to handle most of these via new lexical and phrase structure rules (with delayed triggering and/or inhibitory potentials). An example of the use of lists is seen in *I have four boxcars, b6 at city H, b5 at city F, b7 at city B, and b8 at city I*. This example seems to involve conjunction reduction of the type in (v) above; as is usual, the list itself occupies sentence-final position. Parentheticals in speech are likely to be prosodically marked (corresponding to bracketing by commas, dashes, or dots in text, as in *I think -- correct me if I'm wrong -- there's an empty boxcar at city B*). While in theory parentheticals can be described by GPSG-style metarules, they seem to be another phenomenon (like ellipsis) which requires use of delayed rule triggering to avoid over-prediction. Telegraphic language (and block language, as in headlines) rarely occurs in human-to-human speech, but is necessary to deal with keyboard input. Characteristics are the omission of articles, copulas, and *and*'s. Examples of proscribed colloquialisms are the use of adjectives as adverbs (*He sings real good*) and other abuses (e.g., *So I says to him..., Way to go!*). These could easily be handled in the grammar, with help from the

morphological analyzer, though some distinctive style feature should be used. Under deviant vocabulary we include "verbing" of nouns (*He tricycled away*), and perhaps "massifying" of count nouns and "countifying" of mass nouns. (Note that the preceding sentence itself contains instances of deviant vocabulary in scare quotes) "Massifying" and "countifying" are illustrated respectively by *A year ago they started digging the hole for his house. A year later, there's still more hole than house*; and *How many orange juice will that give us?*. (For some plausible grammatical approaches see Pelletier & Schubert, 1984.) More difficult are creative coinages like *workoholic* (easily understood on first encounter by anyone who knows the meaning of *alcoholic*), but these are not a significant concern in the TRAINS domain. Potentially more important in TRAINS are *ad hoc* abbreviations such as *OJ*, *U of R*, *e's* (for engines), and personal initials. While we have no immediate plans to deal with these "intelligently", we are designing the parser to accept unknown words, tentatively classifying these into multiple alternative categories as a function of their morphology, the parser state, and any previous encounters with the same (unknown) word.

In contrast with these borderline phenomena, we regard the following as outright errors: restarts, various word-level errors, and various character-level errors. Restarts (also called resumptions) are very common and hence particularly important in the analysis of spoken language (e.g., *Ok, now uhh, let me, let me check on the uhh, where the .. where the engines are and the.. the boxcars are uhh*). Here we expect to use (inhibited) phrase structure rules something like *XP* <-- *XP*[*broken-off*] *XP* (with logical form obtained entirely from the second, complete XP on the right), *XP*[*broken-off*] <-- *Y Z*[*omitted*], and *Z*[*omitted*] <-- ..| ... | - | -- | , | *uh* | *uhh*. The second rule (with RHS *Y Z*[*omitted*]) would be delayed so that it is triggered on its second daughter, rather than the first, to avoid over-prediction of restarts. Restarts may be further complicated by occurring over successive turns (in instalments, as above). For instance, in the exchange

<S> I just found city E2
<M> City E2?
<S> uhh engine E2,

the first and third utterances together comprise a sentence with a restarted object noun phrase. In general, therefore, we must allow for piecemeal parsing and interpretation of erroneous input, spanning several turns (interspersed with the interlocutor's comments). Word-level errors include word confusions (e.g., *irregardless*), preposition usage errors (e.g., *Bush ought to have a better grasp on the details* -- see Blejer et al. [1989]), word doubling (e.g., *the the*), word omission (*There two engines at I*), and word contraction and segmentation (*I'll find away*, *I need in formation*). Common word confusions and doubling can be dealt with in the lexicon and grammar; also, the "elastic unification" mechanism readily allows for preposition misuse. Word omission, contraction and segmentation are more problematic and may (in the worst case) require error-driven backtracking.

Other errors include mispronunciations or corresponding errors in keyboard input (e.g., *The train is stationery*) and phoneme omission and insertion (e.g. *He's goin too fast*, *He is lacksadaisical*). Since it will be some time before we will be attempting to handle speech input directly, we expect to have to deal with analogs of mispronunciations in the transcribed keyboard input. Some standard spelling correction techniques should be applicable here, but we will make a minimal effort in this area until we have a clearer agenda for the switch to speech input.

4.4 Semantic Interpretation

We have mentioned our commitment to close coupling of semantic and syntactic theory development, use of a "natural" meaning representation, and modular, compositional design. In particular, we are pairing phrase structure rules one-to-one with compositional rules of interpretation, and striving to keep these interpretive rules simple.

We define rules of interpretation as **compositional** if they express the logical form of a phrase in terms of the LF's of its top-level constituents, without any reference to the *structure* of those constituent LFs. For example, the semantic rule for a declarative sentence simply says "apply the LF of the predicate verb phrase to the LF of the subject noun phrase". Note that a rule like the following would not be compositional in our sense, since it makes reference to parts of the constituent LFs: "If the LF of the noun phrase denotes an ANIMATE entity, create a case frame whose AGENT is given by that LF, and whose event type, tense operator, OBJ, and IOBJ (if any) are inherited from the LF of the verb phrase ...". The compositionality constraint tends to ensure simplicity of the rules. As a further simplifying constraint, we are trying to keep the LF of any uninflected word atomic. (Thus "deliver", as a base-form verb, may be translated logically as "deliver1", "engine" as a singular noun may be translated as "engine2" or "locomotive", etc.) This makes lexical semantics as simple as possible. The LFs of regularly inflected words are obtained by rule, and "knowledge about word meaning" is stored as MPs (in the knowledge base) about *logical translations* of words, rather than as semantic properties of *words*.

The quest for simple, compositional rules of interpretation has had, and continues to have, a large impact on our choice of meaning representation. In fact, if lexical meaning representations are atomic and higher-level rules apply the LF of one constituent to that of the other(s), we are bound to obtain a rather "English-like" sentential LF (such as our Episodic Logic). We prefer the term "natural" meaning representation, to suggest that a representation is not only intuitively close to natural language, but also formally easy to derive from surface syntax and expressively equivalent to natural language. The use of a natural meaning representation is a break with tradition in AI, which has generally been to "shoehorn" sentence meanings into some expressively limited but familiar logic or frame language.

In using a natural meaning representation we are of course following Montague, but our (lexical and nonlexical) semantic rules are generally simpler than Montague's or those in other Montague-like fragments. This is largely because of two reasons: (a) we do not interpret noun phrases as properties of properties of individuals, but rather as simple terms (in the case of proper nouns and pronouns) or unscoped quantified terms (in the case of noun phrases with determiners); besides keeping the logic simpler (avoiding many lambda-abstractions), this also facilitates a simple, uniform treatment of scope ambiguity; (b) in our logic we do not insist that the extension of an expression (its value at an episode, or situation) be a function of the extensions of its parts (as is done in all other intensional meaning representations we know of); instead, we merely require the intension of an expression (a partial function on episodes/ situations) to be a function of the intensions of its parts; this allows us to dispense with Montague's ubiquitous intension and extension operators (and gives us what is called an "inherently intensional" logic in Schubert and Pelletier [1989].

Episodic Logic already has many of the features we desire for a natural, easily computed

meaning representation. We have experimented with fragments of children's stories [Schubert and Hwang 1989; 1990] and with the earlier 8-sentence TRAINS discourse fragment, and these experiments have been encouraging. The TRAINS fragment prompted tentative solutions to such phenomena as sentences expressing necessity or obligation (*We have to make OJ, Shall I start loading...*), intensional verbs of creation (*make OJ* - note that making OJ is not a matter of there being some OJ which is then *made*), the semantics of names (*engine E3*), time and place complements and adjuncts (... *loading the oranges in the empty car at I*, ... *arrive at I at 3pm*), and infinitive complements (... *scheduled to arrive...*). However, our semantically annotated grammar fragments and our experience with the LFs that they produce are still too small to warrant confidence about the stability and adequacy of the logic. Other problematic issues that we foresee are the meaning of a planner "having" a needed resource (*I have two engines to work with*), the meaning of *let's* (*Let's do it, Let's see*, ...), and the meaning of references to strictly nonexistent entities (*the OJ which the factory was supposed to produce*). No doubt many further problems will come to light as we proceed, and significant overhauls of the grammar and Episodic Logic itself are likely to result.

We should also remark that the present quantifier and operator scoping algorithm is very simple-minded, and we plan to adopt more subtle strategies like those in [Hurum 1988]. Making those strategies work on-line in the parser and interleaving them with de-indexing will be a significant challenge.

4.5 De-indexing

The third stage, namely de-indexing using tense trees, is perhaps the most novel aspect of our approach. The branches in a tense tree correspond to *past*, *pres*, *futr*, and *perf* operators, and can be traversed repeatedly as input LFs containing these operators are processed. Typically, each new traversal (or creation of new branches) is accompanied by storage of new episode tokens at the nodes of the tense tree structure. The interpretive rules which give de-indexed LFs make reference to the lists of tokens stored at the nodes. The status of the current implementation is comparable to that of the semantic interpretation stage. We are currently able to deal with many past, present, future, perfect, and temporal PP constructions, using quite simple recursive de-indexing rules for the operators concerned, with allowance for (in principle) arbitrarily deep nesting of operators. However, these rules are still quite tentative and they do not cover all operators which make implicit reference to times, episodes, or situations.

More specifically, the idea is that for each operator there is a pair of rules, the first of which specifies a transformation from a given surface LF and a given tense tree (with a specified *focus*) to a non-indexical LF, and the second of which specifies the corresponding transformation of the tense tree. The result of applying the rules to a new input LF and an initial tense tree, as was illustrated in Fig. 4, is a context-independent (non-indexical) version of the input LF, with explicit episodic variables and temporal relations among these variables which bring to light relations implicit in the tense and aspect operators and temporal prepositional adjuncts and in the surface sequencing of input clauses. The rules work rather well for the small discourse segments we have studied, but undoubtedly will need to be extensively augmented and revised as we consider larger, more varied discourses. Some issues we are aware of are: the semantics of simple past within a clause immediately following a past perfect clause (*The engineer had failed to notice the red light.*

He was very tired from a 16-hour shift.), clausal time adverbials (*The engineer kept driving after he had tried repeatedly to get a replacement*), and noun phrase time reference (*Due to lack of refrigeration, the oranges were now a decaying mass of pulp* -- note that the properties *oranges* and *decaying mass of pulp* apply consecutively rather than concurrently).

We mentioned reference interpretation as an additional function of the de-indexing stage, and our intention to make use of tense trees here as well. The reason this makes sense is that one of the link types in tense trees is an *embedding* link, leading from a tense tree node to the root of an embedded tense tree, where the embedded tree typically corresponds to a subordinate clause. Thus the embedding-link structure of a tense tree structure reflects the clausal configuration of complex sentences, and as such provides a convenient set of "hooks" to which clause-specific context information can be attached. In particular, one can attach clause-specific history lists (i.e., entities referred to) and focus lists (i.e., entities in focus) at these hooks; this would enable application of coreference constraints similar to those used in discourse representation theory. For example, the oddity of *The engineer thought a train was approaching but it wasn't* could be accounted for by such constraints.

The current implementation relies on just a few ad hoc rules for reference interpretation, which need to be replaced by a principled approach. We envisage the local, clause-specific context structure as being the "fine structure" of larger discourse-segment structures like those standardly discussed in the discourse structure literature (e.g., see Allen [1987, ch. 12-14]).

4.6 Input-driven Inference

Finally, the stage of input-driven inference (so far unimplemented) is needed to aid disambiguation, including reference interpretation, to suggest causal connections and other coherence relations, and to make the implicit content of input more explicit via meaning postulates. For this stage we plan to use the approach of Schubert and Hwang [1989;1990], adapting parts of EPILOG, the implemented system based on that approach. Some major research issues here are identifying the kinds meaning postulates and world knowledge (often probabilistically qualified) needed in the TRAINS domain, adjudicating among conflicting or compatible lines of inference bearing on some conclusion, triggering them appropriately, and limiting forward chaining by appropriate criteria.

As an example of inference based on MPs, we mentioned the inference from kinds to instances of those kinds above. Another example would be the inference from *making OJ* to some particular entity having the OJ property (where it did not have that property before). This is crucial in determining the reference of *it* in a sentence like *The factory made OJ and loaded it into tanker cars.* The inference from *A train pulled into the station* to *Probably some people got off the train and some got on* would be based on general world knowledge, and would be crucial to making sense of a subsequent input sentence such as *The crowd converged on the doors of the passenger cars.*

Adjudicating among alternatives uncertain inferences is crucial to finding the most coherent and plausible interpretation of an input. Note for example that depending on prior context, a crowd's surge toward the doors of an arriving train might either be interpreted as being motivated by the intention to board the train, or by the intention of meeting arriving passengers (or catching a glimpse of arriving celebrities, etc.) To the extent that one interpretation has stronger support by prior context and world knowledge, others diminish in credibility. We are exploring abductive

methods similar to those proposed by Hobbs et al. [1988]. These authors view understanding as chaining backward over implications with weighted antecedents towards a "best"explanation of observed data (inputs). However, we think their methods handicapped by the built-in assumption that the only explicit causal knowledge available to the reasoning agent is knowledge leading from causes to effects. Our assumption is that knowledge leading directly from effects to (more or less probable) causes -- much like a doctor's diagnostic knowledge -- will be available as well. This turns explanatory reasoning into *forward* inference, and appears to finesse some logical and computational problems inherent in abduction. Charniak & Goldman [1988] make proposals similar to Hobbs et al.'s, but (appropriately, we think) within a probabilistic framework. They emphasize chaining from subplans (or subparts) to plans (or wholes); however, this is also back-chaining over implications, if plans (wholes) are regarded as implying their parts.

The problem of "triggering" appropriate inferences is important, since failure to make a relevant inference may prevent the intended causal connection, prediction, or interpretation from being discovered; or, if the system is too "trigger-happy", masses of irrelevant rules may be tried and too many inferences generated. We are experimenting with criteria for terminating inferencing, such as decline in the probability and "interest" of the conclusions drawn. Note that the types of inferences under discussion "shade over" into plan-based discourse reasoning (see next section), so that a future task will be to smoothly integrate these forms of reasoning.

5 Discourse Modelling

A focus of the discourse modelling part of the project is the use of planning models to attempt to account for much of the discourse phenomena found in the TRAINS domain. But discourse structure cannot be fully accounted for by general reasoning processes alone, so another part of this project is investigating the structural properties of discourse and attempting to discover how the structural properties interact with the reasoning processes and how the two mutually constrain each other. In this section, we will briefly describe the goals and the approach used to examine these two research areas.

5.1 Dialog Collection

The TRAINS domain was carefully designed so that a significant part of it is within reach of current (or near future) capabilities of plan reasoning systems. Because of this, we hope to be able to fully specify and implement the reasoning underlying the "system" in the dialogs. In addition, the domain was designed so that the dialogs could be quite free-ranging. In particular, the dialogs exhibit complex behavior as both the manager and system take initiative at times, and there are a large number of clarifications, confirmations, negotiations, corrections and acknowledgements.

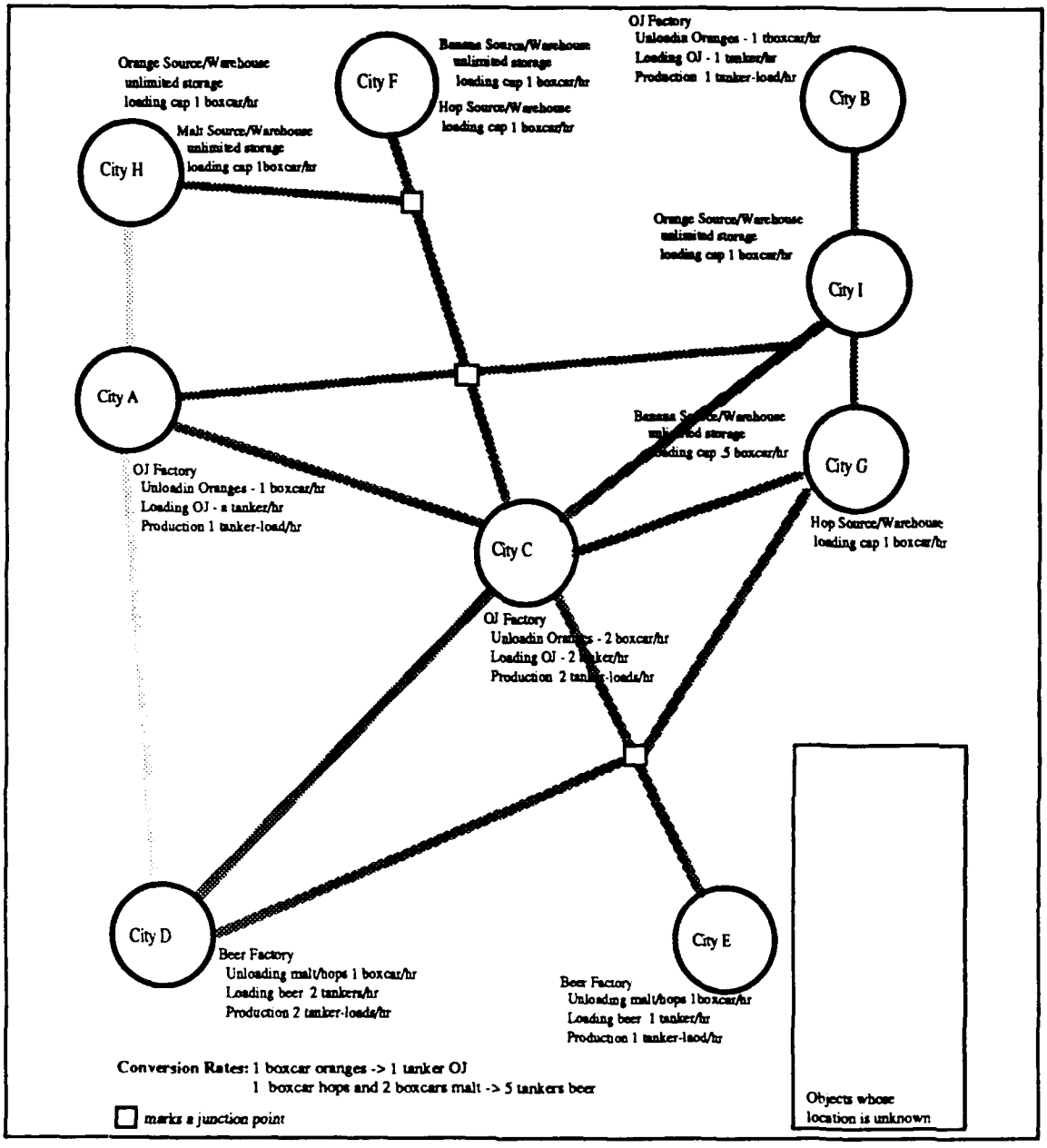


Figure 6: The map for dialog collection

We have collected an initial corpus of natural spoken conversations between two people engaged in complex problem solving in the TRAINS world. One person (simulating the system) has most of the information and detail about the domain, but the other (the manager) has a problem to solve. The two are in different rooms and so have no visual contact, but they both have the same map from which to work, as shown in Figure 6. A fragment from one of the dialogs shown in Figure 7. Each utterance is roughly classified as to its function: whether it is primarily concerned with making progress on solving the problem (plain text), or whether it is primarily concerned with maintaining the conversation itself (in bold). The participants are labelled <M> (for manager) and <S> (for system), although the system here was simulated by another person. Comments on the

<M> **ok, now uhh, let me, let me check on the uhh**
<M> **where the.. where the engines are and the.. the boxcars are uhh**
setting the immediate conversation goals for the following dialog fragment
<M> **I'm assuming,**
indicating that <M> is asking for confirmation
<M> **let's see, that uhh**
<M> is holding the turn while he examines the map
<M> I have two engines to work with, engine E2 which is at city D
<M> and engine E3 which is at city A
<S> **aah,** yes.
the "aah" probably indicates that <S> is thinking about the answer (and acknowledging that the question was understood)
<M> and uh, I've got two tankers, tanker t1 is at city A,
<M> and tanker t2 is at city B
<S> **that's right. hnn, hnn.**
<M> **ok**
<M> acknowledges the reply and indicates that he has accepted <S>'s reply
<S> there're.. there're other tankers as well.
<M> **ok**
<M> acknowledges <S>'s introduction of new information
<S> there're **actually** four tankers at city E
"actually" indicates that <S> believes <M> doesn't know about these tankers
<M> **four tankers at city E, ok**
<M> acknowledges hearing the new information, and then accepts it
<M> **uhh so, tankers t3, t4, t5, and t6 are all at city E.**
<M> confirms his understanding of <S>'s assertions
<S> **that's right**
<S> confirms <M>'s confirmation
<M> **ok.** and just uh
<M> acknowledges the previous exchange and signals a move to a new topic
<M> I have four boxcars, b6 at city H, b5 at city F,
<M> b7 at city B, and b8 at city I.
<S> **that's right.**

Figure 7: An excerpt from the dialogs with the discourse functions identified

possible discourse function of the utterances concerned with maintaining the conversation are presented in italics.

As can be seen, approximately half of the utterances are concerned with maintaining the communication process. There are utterances that identify the goals of the next stretch of discourse, and a large number of utterances that pertain to acknowledging the other participant's utterances and in maintaining a smooth flow of control (i.e. identifying whose turn it is to speak). It has been our claim for some time that this level of discourse interaction must be explicitly modelled if we are to build systems that can converse in natural language. In previous papers we have described a plan-based model that accounted for clarification subdialogs among other things [Litman and Allen 1987; 1990]. We are now attempting to develop an extended model that can account for all the discourse-level interactions found in the corpus.

Using the collected dialogs, we are developing a taxonomy of discourse-level acts that is specific enough so that different people can independently classify each utterance in the same way. We are then using this data to create a database containing a transcription of each utterance annotated by its discourse function. In addition, we are analyzing the actual speech signals to extract prosodic information (primarily pitch contours, speech rate) and adding this information to the database as well. We have started some preliminary studies on prosodic cues to the discourse

acts in our taxonomy, but need to analyze additional data before we have significant results. Our initial efforts along these lines is described in Nakajima and Allen [forthcoming].

Rather than analyze the dialogs in terms of abstract discourse relations, our taxonomy is based entirely on the intentions of the speaker. This allows us to integrate well with previously developed computational speech act models, and provides a slightly different view from the other approaches. It is important to remember that just because a speaker intended an utterance in a certain way, it doesn't mean that the hearer understands it that way. Establishing agreement between the speaker and hearer as to what was intended is the primary reason for acknowledgements, clarifications and corrections. In addition, even if an utterance is understood correctly, this doesn't commit the hearer to accepting the intended consequences of the act (e.g. believing the speaker's assertion, or performing the requested act).

5.2 Speech Act Models

The main intentions underlying an utterance, or group of utterances, is classified by the speech act that the utterance(s) is said to perform. In the TRAINS domain, the principal speech acts are as follows:

REQUEST - the speaker intends to get the hearer to perform some action, and obliges the hearer to respond to the request (e.g. *Take engine E3 to city I*);

SUGGEST - the speaker intends the hearer to consider the suggestion and if the hearer accepts the suggestion, then the speaker is comitted to whatever the suggestion entailed (e.g. *Let's take engine E3 to city I*),

QUESTION - the speaker intends to get the hearer to respond with some information - This includes yes-no questions and wh-questions (e.g. *Where is engine E3?*);

PROMISE - the speaker intends to commit to the promised act if the hearer accepts the promise (e.g. *I'll find out when train E3 will arrive.*)

INFORM - the speaker intends to make a claim about the world (e.g. *Engine E3 is at city A*);

ACCEPT - the speaker accepts the other agent's previous speech act, and thus is comitted what the act entails (such as doing the requested action, accepting the suggestion, believing the inform, and so on);

REJECT - the speaker rejects the other agent's previous speech act, nullifying the intended effects.

Considerable work remains in defining this set of speech acts precisely within a planning model. Most notably, we need a representation of obligation to properly handle promises and offers.

It is important to realize that nearly every speech act can be used at different levels of the conversation: they can involve the plan in the TRAINS world (the **domain level**), or the problem solving process that the two agents are engaged in (the **problem solving level**), or the understanding and managing of the conversation itself (the **discourse level**). Because of the focus on the discourse-level acts in this paper, we will often distinguish these as separately named acts. For example, consider the REQUEST act at the different levels. Two requests at the domain level that are in the transcripts are:

<M> Can you have city I fill B6 with oranges, please

<M> Pick up b8 and take it on up to city B please

At the problem solving level, two requests are:

<M> Let me know when E3 has B6 loaded.

<M> I need some help.

At the discourse level, a request may be a clarification request, as in the following example dialog fragment where the clarification request is in bold italics. The ensuing clarification is also included.

<S> e3 is leaving city H, on its route to city F
<M> with B5
<S> with B5, yes
<S> to city H, that was
<M> that's right

While there is a correlation between syntactic forms and speech act types, the relation is quite complex. In general, the appropriate speech act can only be identified after significant reasoning about how the utterance fits into the current context, especially with what is previously known about the speaker's general goals. Utterances that appear to be one type of speech act can actually be used in different settings for a wide range of different intentions. The utterance *Do you know the secret?*, for instance, can in one setting be a request, in another an offer to tell the secret, in another simply a yes/no question, or a host of other acts. Perrault and Allen [1980] showed that many indirect forms can be derived from the literal interpretation of an utterance by using plan recognition techniques alone. Unfortunately, this approach is insensitive to the way sentences are phrased and thus tends to overgenerate. Hinkelman and Allen [1989] (see also Hinkelman [1990]) extend this work so that phrasing can affect the interpretation, both by suggesting certain "default" indirect readings, and by limiting the range of interpretations that can be derived by plan recognition. These ideas will be tested extensively and extended as necessary in the TRAINS project.

5.3 Utterance Acts

If there were a one-to-one mapping between utterances and speech acts, and all utterances were unambiguous and perfectly understood, then classifying utterances by speech act would be a fairly straightforward task. But this is not the case. In particular, often it takes several utterances and exchanges before a speech act is completed. Consider the following excerpt from one of the dialogs that involves making a suggestion.

<M>	*What if we would stop it uhh*	Suggestion started
<M>	*pull uhh those tankers of beer to city G*	Suggestion continued
<S>	*hnn-hnn*	Acknowledgment
<M>	*and then from city G, leave the tankers there*	Suggestion continued
<S>	*yes*	Acknowledgement
<M>	*goto city I*	Suggestion completed
<S>	*right.*	Acceptance of the suggestion

It takes the manager four utterances to make the suggestion, and the system acknowledged parts of the suggestion twice before it was completed and accepted. Yet this whole subdialog needs to be analyzed at some level as a single suggestion since the acceptance applies to the entire subdialog.

We are investigating models to handle such phenomena by positing a more detailed level of interaction in terms of utterance acts. At this level, an utterance may initiate a speech act, continue a speech act, or correct what has been said so far. In addition, the other agent may acknowledge the act so far, or request clarification or correction, and so on. Once the speech act appears to be mutually understood between the two agents, it may then be accepted or rejected. We are currently exploring models of this process and how the utterance act level relates to the speech act level.

When clarification requests are introduced, the structure of the dialog becomes even more complex as entire sub-dialogs can occur in order to clarify the intent of the containing dialog. This introduces a hierarchical structure to the dialogs as has been proposed by many (e.g. see Grosz and Sidner [1986]). In addition, a dialog may be interrupted and later resumed as in the following example, where indentation is used to indicate the segmentation structure:

<M> mean time we send E2 over to city I	Suggestion initiated
<S> ohoops, hhhh. yes, one moment - I have to find engine E2 somewhere	Digression Introduction
<M> Its in city D right now, according to my best information	Response to digression
<M> But you're the system	Continue response
<S> I just found city E2	Continue digression
<M> City E2?	Clarification request
<S> uhh engine E2	Correction
<S> that was to city A ?	Clarification Request
<M> ahh city I	Correction
<S> to city I	Confirming correction
<M> yes	Acknowledge
<S> ok	Accept/Acknowledge completes the digression
<M> and I guess while that's going on ...	Continuation of first utterance

This digression shows the need to be able to handle shifts in topic that are not a natural progression or decomposition of the previous topic.Within the digression we see clarification requests, corrections and acknowledgements. Finally, after the digression is completed, the first sentence is continued that though the digression never occurred. This example shows strong evidence for a stack-like structure of discourse topics as suggested by Grosz and Sidner and others. The model of Litman and Allen (1990) used such a model within the planning framework and suggests a way to integrate discourse processing and plan reasoning. The model, however, will need significant extension before it can handle complexities such as those that arise in this example. We believe that a closer analysis of the level of utterance acts may provide the framework in which such complexities can be handled.

A speech act is realized by one or more utterance acts, beginning with an initiating utterance, followed by possible clarifications and corrections, clarification requests, partial acknowledgments and so on. The speech act is completed by an implicit or explicit acknowledgement. Acknowledging a speech act does not commit one to accepting the consequences of the act, it simply indicates that the speech act is understood. Thus the entire utterance level analysis is driven by this process of establishing mutual belief about what speech act was just performed. Directly

relevant to this research is work by Clark and his colleagues on establishing a common ground (e.g. [Clark and Schaefer, 1989]) and work in Conversational Analysis (e.g. [Sacks et al, 1974; Schegloff et al, 1977]).

5.4 Discourse Modelling and Reasoning

The discourse module must use the structural properties of the dialog and the domain plan reasoner, to maintain the state of the dialog. For the most part, the state of the dialog is maintained by keeping track of what information the two participants have proposed, what has been understood and what has been agreed upon. In addition, the discourse model must maintain structural information such as focusing and centering information to deal with problems such as reference. In this section, we will concentrate only on those aspects of the model that interact with the domain plan reasoning.

The domain plan reasoner uses planning and plan recognition techniques to maintain the current state of the domain plans under discussion. See Ferguson [1991] for more details on the current system. The discourse reasoner calls the domain reasoner to verify hypotheses about the discourse function of the utterances, and to update the state of the plan as needed. Plan fragments in the knowledge base must be characterized by *at least* six different modalities to capture the state of the discourse. These are organized hierarchically with inheritance so that we can examine the full plan from either the manager's or the system's perspective as shown in Figure 8.

The modalities include:

- the plan fragment suggested by the manager but not yet acknowledged by the system (Manager-Proposed-Plan-Private);
- the plan fragment suggested by the system and not yet acknowledged by the manager (System-Proposed-Plan-Private);
- the plan fragment suggested by the manager and acknowledged but not yet accepted by the system (Manager-Proposed-Plan);
- the plan fragment suggested by the system and acknowledged but not yet accepted by the manager (System-Proposed-Plan);
- the plan fragment that is shared between the two (i.e. accepted by both) (Shared-Plan); and
- the plan fragment constructed by the system but not yet suggested (System-Private-Plan).

Each context is associated with a particular form of plan reasoning as indicated in the figure. In particular, the plan in the System-Private-Plan context is extended by plan construction (essentially classical planning), where the plans in all the other contexts are extended by plan recognition relative to the appropriate set of beliefs.

Figure 8 also shows how plan fragments may move between the various contexts. A suggestion from the manager enters a new plan fragment into the Manager-Proposed-Plan-Private context and initiates plan recognition with respect to what the system believes about the manager's private beliefs. Once acknowledged, this suggestion becomes "public" (i.e. it is in Manager-Proposed-Plan). An acceptance from the system would then move that plan fragment into the Shared-Plan context, again invoking plan recognition.

Planning by the system results in new actions in the System-Private-Plan context. To make these actions part of the Shared-Plan context, the system must suggest the actions and then depend on the manager to acknowledge and accept them. This model, while still crude by philosophical

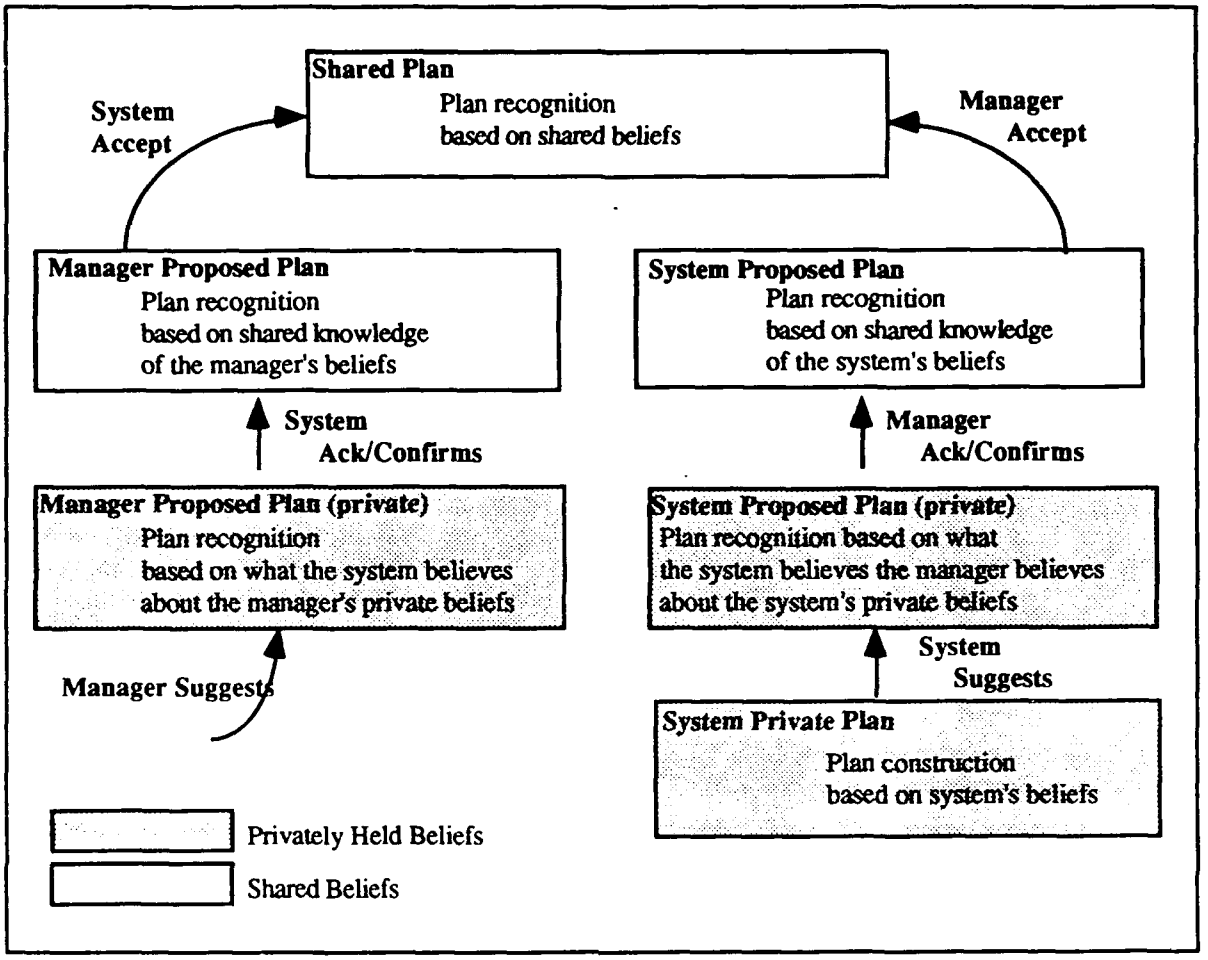


Figure 8: The different plan modalities from the system's perspective

standards, it seems rich enough to model a wide range of the discourse acts involving clarification, acknowledgment and the suggest/accept speech act cycle abundantly present in dialogs in this setting.

Because of the inheritance through the spaces, when the system is planning in the System-Private-Plan context, it sees a plan consisting of all the shared goals and actions, what it has already suggested, and all the new actions it has introduced into the plan privately but not yet suggested.

Consider a simple example. Assume that the Shared-Plan context contains a plan to move some oranges to a factory at B, but there is no specification of the engine to be used. The system might plan to use engine E3. At this stage, the plan from the System-Private-Plan context involves E3. The plan in the System-Proposed context, however, is still the same as the plan in the Shared-Plan context, which still does not identify which engine to use. When the system makes the suggestion, the plan fragment involving E3 is added to the system proposed plan (private). An acknowledgment from the manager results in this plan fragment being added to the system-proposed plan known to both agents. If the manager then accepts this, it then becomes part of the shared plan. If the manager rejects the suggestion, then E3 does not become part of the shared plan (at least, not without further discussion). This same type of scenario arose in the sample dialogs in

the TRAINS-90 system when the system suggested using a particular car to use to move the oranges.

The prototype system uses a simple model along these lines and can handle examples in which the two agents are free to accept or reject suggestions as they are made in the dialog. It is described in more detail in Traum [1991]. The system under development will extend the current one to support acknowledgement behavior, plus some forms of negotiation between the agents in order to arrive at a mutually agreeable plan.

6 Summary

The TRAINS domain is designed to balance the complexity of the planning task with the complexity of the linguistic behavior. Too simple a planning domain would not allow rich dialogs, while a too complex planning domain would not be implementable. The planning/scheduling domain is sufficiently difficult to push planning research, yet within range of producing a reasonable prototype system within the next five years. The linguistic phenomena that arise, on the other hand, are highly complex and do not appear seriously constrained by the domain. As such, it is an ideal research testbed for both natural language work and planning work. Our goal is to produce, within a five year time span, a system with enough functionality that a naive user could be given a problem in the TRAINS world and actually use the system to produce a solution. To realistically approach this goal, we feel it is important to have a working system at all times, however limited it might be. TRAINS-90 was the first iteration of this process, and we plan to produce a new system each summer throughout the five year time period, each year increasing the capabilities and robustness of the system.

The TRAINS project is only in its initial stages, and this report describes our best attempt to describe the current approaches and future goals from a standpoint of less than one year into the project. As a result, much of what is proposed here will change significantly in the coming years. Our progress will be documented with a series of technical notes, starting with the current set that describes the system as of the Fall, 1990.

Acknowledgements

The TRAINS project was explored and refined in the summer of 1990 with great help from George Ferguson, Janet Hitzeman, Chung Hee Hwang, Alice Kyburg, Marc Light, Nat Martin, Brad Miller, Shin'ya Nakajima, David Traum, and Josh Tenenberg, who all worked on defining and building the TRAINS-90 system. The results of this effort are documented in the series of TRAINS technical reports listed in the references.

References

Allen, J.F. & Perrault, C.R. Analyzing intention in utterances, *Artificial Intelligence* 15, 1980.

Allen, J.F. (ed) *American Journal of Computational Linguistics 9*, no.'s 3-4, "Special issue on ill-formed input", 1983a.

Allen, J.F. Maintaining knowledge about temporal intervals *Comm. ACM* 26(11), 1983b.

Allen, J.F. and Koomen J.A: Planning using a temporal world model, *Proc. of the Int. Joint Conf. on Artificial Intelligence (IJCAI 83)*, 1983.

Allen, J.F. Planning as temporal reasoning, in *Proc. of the Int'l Conference on Knowledge Representation and Reasoning*, Morgan Kaufmann, 1991.

Allen, J.F., Kautz, H., Pelavin, R. and Tenenberg, J. *Reasoning about Plans*, Morgan-Kaufman, 1991.

Bacchus, F. *Representing and Reasoning with Probabilistic Knowledge*, Ph.D. thesis, University of Alberta, 1989.

Bacchus, F. *Representing and Reasoning with Probabilistic Knowledge*, MIT Press, 1990.

Blejer, H.R., Flank, S., and Kehler, A. On representing governed prepositions and handling "incorrect" and novel prepositions, *Proc. of the 27th Ann. Meet. of the ACL*, June 26-29, Vancouver, B.C., pp. 110-117, 1989.

Briscoe, E. J., Grover, C., Boguraev, B. K., and Carroll, J. A., A formalism and environment for the development of a large grammar of English, *Proc. of the Int. Joint Conf. on Artificial Intelligence (IJCAI 87)*, Aug. 23-28, Milan, pp 703-708, 1987.

Charniak, E., and Goldman, R., A logic for semantic interpretation, Proc. of the 26th Ann. Meet. of the ACL, June 7-10, Buffalo, NY, pp. 87-94, 1988.

Clark, H. and Schaefer, E., Contributing to discourse, *Cognitive Science* 13, pp. 259-294, 1989.

Cohen, P.R. & Perrault, C.R. Elements of a plan-based theory of speech acts, *Cognitive Science, 3*, 1979.

Dean, T. and K. Kanazawa. Probabilistic temporal reasoning, *Proc. National Conf. on Artificial Intelligence*, pp. 524-528, 1988.

Dean, Thomas and McDermott, Drew: Temporal data base management *Artificial Intelligence 32* pp. 1-55, 1987.

Doyle, Jon, A truth maintenance system, *Artificial Intelligence 12*, pp. 231-272, 1979.

Durham, I., Lamb, D.D., and Saxe, J.B., Spelling correction in user interfaces, *CACM 26*, 1983.

Ferguson, G. *Domain Plan Reasoning in* TRAINS-*90*, TRAINS Technical Note 91-2, Computer Science Dept, University of Rochester, 1991.

Grosz, B. and Sidner, C. Attention, intention, and the structure of discourse, *Computational Linguistics 12*, 3 pp. 175-204, 1986.

Halpern, J. An analysis of first-order logics of probability, *Artificial Intelligence 46*, 3 pp 311-350, 1989.

Harrison, P. and Maxwell, M. A new implementation for GPSG, *Proc. of the 6th Can. Conf. on Artificial Intelligence (CSCSI-86)* Ecole Polytechnique de Montreal, Quebec, pp. 78-83, 1986.

Hinkelman, E and Allen, J.F. Two constraints on speech act ambiguity, *Proc. of the 27th ACL*, UBC, British Columbia, Canada, 1989.

Hinkelman, E. *Linguistic and Pragmatic Constraints on Utterance Interpretation*, Phd thesis, Computer Science Dept, University of Rochester, 1990.

Hobbs, J.R., Stickel, M., Martin, P., and Edwards, D., Interpretation as abduction, *Proc. of the 26th Ann. Meet. of the ACL*, June 7-10, Buffalo, NY, pp. 95-103. A revised and expanded version (with co-author D. Appelt instead of D. Edwards) appears as SRI Tech. Note 499, SRI International, Menlo Park, CA, 1990.

Hurum, S. Handling scope ambiguities in English, *Proc. 2nd ACL Conf. on Applied Natural Language Processing*, Austin, TX. pp 58-65, 1988.

Jensen, K. and Heidorn, G.E. The fitted parse: 100% parsing capability in a syntactic grammar of English, *Proc. of the Conf. on Applied Natural Language Processing*, Feb. 1-3, Santa Monica, CA, pp. 93-98, 1983.

Kautz, H and Allen, J. Generalized plan recognition, *Proc.Nat. Conf. on Artificial Intelligence* , 1986.

Kautz, H. *A Formal Theory of Plan Recognition.* PhD thesis, University of Rochester, Dept. of Computer Science, Rochester, NY, 1987.

Kyburg, H. Objective probabilities, *Proc. of the Int. Joint Conf. on Artificial Intelligence (IJCAI 87)*, pp. 902-904. 1987.

Light, M. *Semantic Interpretation in* TRAINS-*90*, TRAINS Technical Note 91-3, Computer Science Dept, University of Rochester, 1991.

Litman, D.J. & Allen, J.F. A plan recognition model for subdialogs in conversation, *Cognitive Science 11*, 1987.

Litman, D.J. & Allen, J.F. Discourse processing and commonsense plans, in *Intentions in Communication*, P. Cohen, J. Morgan and M. Pollack (eds), MIT Press, 1990.

Loui, R. *Theory and Computation of Uncertain Inference and Decision*, Phd thesis, Univ. of Rochester, Dept. of Computer Science, 1987.

Martin, N and Miller, B. *The* TRAINS-*90 Simulator*, TRAINS Technical Note 91-4, Computer Science Dept, University of Rochester, 1991.

Mellish, C.S., Some chart-based techniques for parsing ill-formed input, *Proc. of the 27th Ann. Meeting. of the ACL*, June 26-29, Vancouver, B.C., pp. 102-109, 1989.

Miller, S.A. and Schubert, L.K., Time revisited, *Proc CSCSI*, 1988.

Nakajima, S. and Allen, J.F. A study of pragmatic roles of prosody in the TRAINS dialogs, TRAINS Technical Note, Computer Science Dept, University of Rochester, forthcoming.

Pearl , J, *Probabilistic Reasoning in Intelligent Systems*, Morgan Kaufmann, Palo Alto, CA, 1988

Pelletier, F.J., and Schubert, L.K., Two theories for computing the logical form of mass expressions, *Proc. 10th Int. Conf. on Comp. Ling. (COLING 84)*, July 2-6, Stanford U., pp 108-111, 1984.

Perrault, C.R. and Allen, J.F. A plan-based analysis of indirect speech acts, *American Journal of Computational Linguistics, 6:3*, pp167-82, 1980.

Sacerdoti, E *A Structure for Plans and Behavior* American Elsevier, 1977.

Sacks, H., Schegloff, E., and Jefferson, G. A simplest systematics for the organization of turn-taking for conversation, *Language* 50, pp. 696-735, 1974.

Sag I.A., Gazdar G., Wasow, T., and Weisler, S., Coordination and how to distinguish categories, *Natural Language and Linguistic Theory 3*, pp. 117-171, 1985.

Sager, N. *Natural Language Information Processing: A Computer Grammar of English and its Applications*, Addison-Wesley, Reading, MA, 1981.

Schegloff, E., Jefferson, G. and Sacks, H. The preference for self correction in the organization of repair in conversation", *Language* 53, pp. 361-382, 1977.

Schubert,L.K. and Pelletier, F.J. From English to logic: context-free computation of conventional logical translations, *Computational Linguistics, 8*, pp26-44, 1982.

Schubert, L.K., Are there preference trade-offs in attachment decisions? *Proc.Nat. Conf. on AI (AAAI-86)*, Aug. 11-15, Philadelphia, PA, pp. 601-5, 1986.

Schubert, L.K. and Hwang, C.H. An episodic knowledge representation for narrative texts, *First Int'l Conf. on Principles of Knowledge Representation and Reasoning*, Toronto, Canada. Morgan Kaufmann Pub. Co., 1989.

Schubert, L.K. and Hwang, C.H. Picking reference events from tense trees, *Proc. of the DARPA Speech and Natural Language Workshop.* Hidden Valley, PA, 1990.

Schubert, L.K., and Pelletier, F.J, Generically speaking, or, using Discourse Representation Theory to interpret generics, in Chierchia, G., Partee, B.H., and Turner, R. (eds.), *Properties, Types and Meaning II*, Kluwer, Dortrecht, pp. 193-268, 1989.

Schubert, L.K. Monotonic solution of the frame problem in the situation calculus: an efficient method for worlds with fully specified actions, in H.E. Kyburg, Jr., R.P. Loui, and G.N.Carlson (eds.), *Knowledge Representation and Defeasible Reasoning*, Kluwer, Dortrecht/Boston/London, 23-67, 1990.

Schubert, L.K. *Language Processing in the* TRAINS *Project*, TRAINS Technical Note, Computer Science Dept, University of Rochester, forthcoming.

Shoham, Y. *Reasoning and Change*, MIT Press, 1988.

Stefik,M Planning with constraints - MOLGEN: Part 1, *Artificial Intelligence 16(2)* pp. 111-139, 1981.

Tenenberg, J.. *Abstraction in Planning*, PhD thesis, University of Rochester, Dept. of Computer Science, Rochester, NY, May 1988.

Traum, D. *The Discourse Reasoner in* TRAINS-*90*, TRAINS Technical Note 91-5, Computer Science Dept, University of Rochester, 1991.

Vere, S. A: Planning in time: windows and durations for activities and goals *PAMI 5*, pp 246-267, 1983.